Report No. UIUCDCS-R-74-653

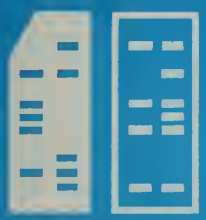
# INTERACTIVE EXPRESS STATISTICAL SYSTEM

by

William Charles Walter

June 1974





DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

Report No. UIUCDCS-R-74-653

# INTERACTIVE EXPRESS STATISTICAL SYSTEM

BY

William Charles Walter

June 1974

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

# ACKNOWLEDGEMENT

I would like to express my gratitude to my advisor Professor Richard G. Montanelli and to the staff of the Statistical Services Office at the University of Illinois. Their comments and criticisms were a very valuable aid to me during the design, testing, and documentation phases of the interactive prompter.

# TABLE OF CONTENTS

# Chapter 1 -- Introduction

In this introduction chapter, I would like to acquaint you with the computational environment at the University of Illinois which, I believe, will be enhanced by the implementation of an interative statistical package. I will start by covering a few general areas for those who may not be familiar with the local facilities. This will lead into a discussion of data processing as viewed only from the soft science disciplines. What computational benefits are provided for the psychologist or econometrician? Do local features or systems exist which are tailored to their requirements? My approach will point out a class of users who have a need for performing elementary statistical computations. It will also touch upon the system resources which could be utilized to answer this need. Presently there are statistical packages which can get the job done. The question is whether they provide a maximum of ease and efficiency for the user.

The University of Illinois maintains a sophisticated set of computing facilities spread across the state at several campuses. At the Urbana-Champaign campus, the research and educational data procesing facility is governed by the Computing Services Office (CSO). The main computer facility consists of an IBM 360/75 with a megabyte of high speed memory and two megabytes of large capacity storage enhanced by a full complement of drum, disk, and tape storage units. Operating under OS-MVT release 20.6 and HASP, over 600K is allocated to a constant stream of batch jobs which run in a variable size partition, multiprogrammed environment using seven initiators. The remainder of high speed core and some large capacity storage have fixed size partitions with initiators allocated to supervisory routines, on-line application programs, our local time-sharing system, and an express job system. The combination of the latter two systems provide the features necessary for the Interactive Express Statistic System (IESS). This will be discussed in detail below.

This data processing facility at University of Illinois, as with many other universities, evolved from the Digital Computer Lab and its

evolution from within engineering and mathematical application areas. Until recent years, introductory courses were directed to and required by only the math, physics, and engineering departments. The majority of research time was also allocated to users in these areas. Even though many segments of the research community in the soft science areas have been utilizing the computer for twenty or so years, a real awareness and course offering in these areas has not emerged at the University of Illinois until recent years. Now, due to this awareness and the ever increasing number of research producing computational techniques used in these areas, a rapidly expanding application area has developed. So great a need existed at the University of Illinois, that a separate programming and consulting area was established around CSO's main remote terminal area in the liberal arts and commerce area of campus. The main functioning unit of this new area is now known as the Statistical Services offices. They provide programming, consulting, research and coordinating functions for statistical applications in a variety of disciplines.

The main statistical system supported by the Statistical Service area is called SOUPAC (Statistical Oriented Users Programming and Consulting). It was developed through the efforts of the late Professor Kern Dickman and the SOUPAC staff, which has been functioning on campus since the 1950's and became the nucleus of the Statistical Services group. The SOUPAC statistical system presently consists of over seventy statistic procedures programmed in Fortran. They are accessed sequentially by an assembler language executive monitor and utilize a common I/O subsystem, also programmed in assembler language, and resident in a local SOUPAC subroutine library. Some of the most used statistical techniques in SOUPAC are regression analysis, frequency counting, analysis of variance, factor analysis, and econometrics programs. The usage of the other routines range from frequent to almost never. SOUPAC has power and flexibility not often found in other statistical packages.

There are always trade-offs involved when using a large general purpose application package. The data manipulation and statistical capabilities of SOUPAC are almost unlimited. The availability of these

and many other features has made SOUPAC large and cumbersome for some applications. Even the simplest analysis requires a region of over 120K. Inputting and outputting of large files can be slowed down by the general I/O subroutines used to provide a common data handling interface among all the different routines. The arguments for and against general purpose systems are analogous to arguments in computer design concerning the general purpose machine versus the special purpose machine. The general purpose machine can do almost everything reasonably well, but nothing as well as the specially designed machine for the particular application.

In response to this need, a small stripped down version of SOUPAC was developed which contains several of the most frequently used statistical routines but lacks most system features and data manipulation capabilities of SOUPAC. This package, called SOUPEX, was designed to run on the previously mentioned expressed job system. The express system has a constantly allocated partition and initiator, and uses a dedicated card reader and printer to provide almost instant turnaround time for jobs with very low time, lines, region, and I/O requirements. A maximum of thirty variables limitation was placed on any analysis run with SOUPEX. Also, no parameter specifications were provided to enable one to specify options as existed in SOUPAC. These requirements are limiting to the general purpose user, but for a variety of users, instant turnaround and ease of programming has been provided. Teachers, who assign exercises in statistical analysis, can tailor their assignments to be run on SOUPEX. Researchers searching for preliminary descriptive statistics on their data can use SOUPEX. Almost every advanced user requires small and simple analyses at certain stages of his work, which could be performed by SOUPEX. The express system becomes even more beneficial during peak load times during the semester, when the shortest turnaround for a batch job is often longer than four hours.

Another facility on campus which is beneficial to many classes of users is the University of Illinois PLORTS Time-Sharing system. As with the express job system, PLORTS maintains an initiator with constant partitions in high and low speed core. Using the 360 in slave mode are

two front-end processors, a PDP-7 and a Microdata. These processors, in turn, can be accessed via teletypes, IBM 2741 terminals, and Hazeltine display terminals. In the future, only the Microdata will be used. The system is presently in a conversion period. The basic features of the PLORTS Time-Sharing system include on-line file editing, job entry into the batch job queue (HASP) or into the express job system, an on-line calculator program, and the recent addition of Basic and Fortran languages in interactive mode. More advanced users with easy access to PLORTS terminals can keep their programs and data in PLORTS files. They can be modified easily using the file editing procedures and submitted to HASP or Express just as if they had been read through a card reader. Also, the printed output can be routed back into the PLORTS filing system where it can be conveniently listed. The above procedures take much of the leg work out of submitting jobs and returning later to retrieve the output. If you are using an application which is available on the express job system, you can virtually achieve instant turnaround at your terminal. At a single sitting you can accomplish what would before have required many trips to the routing room and at the least several hours of waiting.

Unfortunately, the naive user is not proficient with PLORTS editing features and other skills necessary to easily undertake the above procedures. They are not difficult to learn, but seem awesome to many and not worth the effort for other users. It is just this class of users who would benefit most from an easy to use, self-teaching interactive application program which would help him set up his job stream, submit it for execution, and display the results to him at the terminal.

In order to justify the energy in creating such an application program, it would have to be directed to a large community of users whose experience-level would welcome this aid. The community of users in obvious need of this service in my application areas include, for example, the student of an economics, psychology, or advertising class whose teacher wanted him to have experience in running a statistical application. So often the teacher briefly explains the theory involved and then, after writing a few control cards on the board, sends his

student off to the computer for the first time. Also in this plight is the graduate student, who after gathering his data, wants to perform a few statistical analyses, but lacks knowledge of computer usage. These users usually seek help from the consultants at the Statistical Services Office. There, instead of undergoing a learning process and benefiting from a hands-on experience with the computer, often the consultant does most of the thinking and the user only acts as a messenger between the card reader, printer, and consultant. No negative reflection on the consultants is intended, as it is not their job to be educators, nor do they have the time, when others are waiting for their services. For the students, this type of experience is often frustrating and does not provide much educational benefit.

These were only some of the many impetuses which encouraged the development of an interactive statistical application package which could be accessed through the PLORTS Time-Sharing system. It wasn't until late 1973 that this became conceiveable. At that time, interactive Call-OS Fortran was implemented on PLORTS with features that enabled jobs to be submitted to and retrieved from the express job system. Before that time all computation was limited to the PLORTS time-slicing environment, which would have been very inefficient for CPU bound statistical computations. It was at this time that the development of the interactive statistical prompter for the SOUPAC Express Statistical System was undertaken. It would be designed for class users and researchers whose computational needs or lack of computer experience would not require the use of the large general purpose SOUPAC statistical system.

The advantages in using the prompter will include almost instant turnaround time and the convenience of working through a terminal. Having more jobs routed through the express job system, would be one more step in reducing the heavy backlog of batch jobs. The final far-reaching goal of the interactive statistical prompter is to have the prompter asking and answering many questions arising in the mind of the user. Hopefully, with a minimum of direction from his teacher or consultant, he will be able to follow his statistical analysis through to completion. This will be a more beneficial experience for him and

will enable the statistical consultants to allocate their time more efficiently.

# Chapter 2 -- Design Considerations

There are many factors which affect design of an on-line application program. What function is the program to serve? With what class of users is it going to be dealing? Where do the trade-off points exist where more features are only more code and not benefits to the user? Beyond these subjective design considerations often lurk hardware and software limitations which further compromise the final product. Many road blocks cannot be anticipated until one has proceeded in the wrong direction a few times. On the other hand, advantageous features may not be recognized until the time where a non-trivial change would be required to include them in the prompter design. Just as the prompter is intended to be of educational benefit for its users, its design and implementation has served the same purpose for its creator.

In order to review the decisions which contributed to the final design, a brief explanation of parts of SOUPEX, Call-OS Fortran, and the PLORTS-Express interface will be necessary. Most of the critical design considerations were due to the software environment created by their interaction.

Before being modified to be compatible with the prompter, SOUPEX had only two types of input specifications. These were a sequence of programs to be executed and a data deck. Neither parameters, keywords, nor options were allowed. The user had available only a strict prescribed analysis. The following example performed a factor analysis and varimax rotation by executing the three programs CORrelations, PRIncipal axis, and VARimax.

```
// EXEC SOUPEX
COR. PRI. VAR. END
DATA (4) (4F2.0)
- data cards -
END#
```

There was no way for the user to specify options such as which matrix to factor, what percent variance to be removed, the number of factors to be extracted, or what items to included in the printout. Clearly, with these limitations, SOUPEX could not support an interactive prompter.

The initial modification made to SOUPEX consisted of the addition of a trivial program which sets a logical variable if SOUPEX is accessed by the prompter. When this logical variable is set, SOUPEX assumes the existence, in preset places in the job stream, of sets of binary information consisting of predecoded parameter information. This now allows a variety of options in SOUPEX runs on express, as long as the job stream is created by another program with the ability to translate the users wishes into binary coded information and the ability to initiate express job runs. Both these functions can be handled by an interactive program executing on PLORTS.

In comparison to the above example, the job stream submitted by the prompter to perform the same analysis with parameter information included would appear as follows:

```
// EXEC SOUPEX
STONEY.  COR.  PRI.  VAR.  END
- 2 binary parameter cards -
DATA (4) (4F2.0)
- data cards -
END#
- 4 binary parameter cards -
```

The STONEY program sets the prompter logical variable which causes all programs to assume the existance of binary parameter cards. The CORrelations program reads its two parameter cards and then the data. Its output is automatically passed to the PRIncipal axis program, which reads two parameter cards before performing its part of the analysis. Finally, the VARimax rotation program reads its two parameter cards and concludes the analysis.

The interface between PLORTS and Express is very simple. While executing an interactive Call-OS Fortran job, a series of subroutine calls can be used to establish a job stream. They are followed by another subroutine call which submits the job stream to the Express job queue. The only way information can be returned to PLORTS is to have the entire output filed back into a PLORTS file. In the meantime, the interactive Fortran job has issued a subroutine call which puts itself into a wait state until the execution and filing process are completed.

There are two alternatives to retreiving information from the output. The program can open the output file; read it sequentially; perform its own parcing and decision making process; extract the relevant information; format it for output; and then display it to the user. The other alternative consists of calling a print subroutine which displays the contents of the file, either in entirety or contiguous subsets. It would clearly seem advantageous to organize the SOUPEX output so the second alternative would be possible.

The second modification to SOUPEX modified the output returned to PLORTS files to be more readable to the interactive Fortran print subroutine. As with the initial modification, this feature is only put into effect if the prompter logical variable is set. To make the output readable and to allow small sections to be independently displayed to the user, a series of print flags are placed in strategic parts of the output. These flags are coordinated with tables in the prompter. When a user wants to see a certain section of his output, the proper flags are supplied to the print subroutine and the correct section is displayed.

The greatest software influence to the design of the prompter came from the limitations in the language in which it was programmed. Call-OS Fortran is, in general, a subset of IBM Fortran IV. It lacks logical variables, logical operators, named common, entry points, block data, direct access I/O, and some other features available in Fortran IV. Features in Call-OS Fortran not available in Fortran IV include free-form statement formats, easier statement continuation procedures, a set of subroutines for file access, and job entry capabilities to HASP or Express.

The following program limits of Call-OS Fortran had a great effect on the prompter.

| | |
|---|---|
| number of source lines in program | 800 |
| number of source characters | 28848 |
| number of bytes of literal storage | 2048 |

When designing a system, it is important to provide for the capability of future expansion. It might have been possible to implement some kind of prompter with a single program under these limits, but no room for addition of new statistical analyses or new features could have been provided. It would indeed have been a compromised project.

For this reason, the existing statistical programs in SOUPEX were divided into logical sets. A separate interactive prompter was to be constructed for each set. I will refer to each of the separate prompters as a statistical section of the Interactive Express Statistical System (IESS). The original four statistical sections were decided to be correlations- factor analysis, regression analysis, linear programming, and one-way frequency counts.

With the distinct statistical section philosophy established, the emphasis now centered on the design of a standard building block which could be used as the base structure for all the above statistical sections and new ones to be added later. The block design would still be restricted by some Call-OS Fortran limits, but the future looked much brighter than before. The proper design of the statistical section building block should lend itself to almost any interactive application. Therefore, after the first statistical section is completed, the implementations of others should only involve insertion in the proper slots of messages, responses, flows, and segments of code for the particular application.

The single most important part of an interactive program is the sequence and level of messages used to communicate with the user. If they are too lengthy and in detail, the experienced user will soon grow tired of waiting and will sense an inefficient use of his time. If they

are too short and abbreviated, then the new user will soon become confused. The prompter should be designed for a wide variety of user levels. Since it is impossible to develope a single vocabulary at the perfect level for all users, the prompter must be capable of operating in a different mode for each class of user. The decision was made to divide the communications with the user into three distinct levels. The three message levels would be directed towards the beginning, the familiar, and the experienced user. Naturally the beginning message level would have lengthy descriptions concerning what was being asked. The experienced level would give only concise, abbreviated messages. The familiar level is the default when executing an IESS program. It is worded using complete sentences with no extra wordiness. If a user has trouble understanding the familiar level, he can type 'HELP' and the message will be repeated in the more detailed beginners level. If he desires the short message level, he can type ' EXPERIENCED '. All messages will be displayed in the short form until he types 'HELP' to return to the familiar level.

The next decision in the statistical section design concerned the method of displaying messages. What first comes to one's mind is to use standard Fortran write statements and include the messages as literals in format statements. Another possibility is to have arrays filled with characters and then write the arrays with alphanumeric formats. The problem with either of these approaches becomes obvious when viewing the Call-OS Fortran limits. Consider forty mesages each written in the three user experience modes. If they only average twenty characters apiece, the literal storage limit of Call-OS Fortran would already be exceeded. This doesn't even include the literal storage required for the table of valid responses and error recovery messages. Clearly, an alternative method had to be used.

Since there already existed a Call-OS Fortran subroutine which displayed the contents of a file between two flags, it was decided to put all the messages into PLORTS files. They could be accessed and displayed using a method similar to the one used to display sections of output to the users. All three levels of each message would be stored sequentially in a file. Each message would begin and end with a

separate eight character flag. The flags associated with each message could be calculated using the message number and experience level. A table of flags stored in each statistical would not be necessary. All messages would be displayed by the same subroutine call, with the flags for the particular message being supplied as parameters. Using this method, no limit would have to be placed on the number of messages used to communicate with the user.

A separate message file could exist for each statistical section. Also, a common message file could be created to contain messages which appear in all of the statistical sections. The common set of messages would be used when refering to functons such as data handling, submission of jobs, viewing or destroying files, listing printed copy of outputs, and other functions which are independent from the type of statistical analysis being performed. Another advantage in having many special purpose message files is the reduction of search time during execution. A message file is scanned sequentially until the starting flag is found. Having several shorter files reduces the average scan time.

The basic design of each statistical section calls for an almost completely table driven environment. There are three basic driver tables, each containing an entry or set of entries for each message. They are called the message type table, the flow table, and the response table. There are two types of messages. A type 1 message requires a response which exists in a set of valid responses. A type 2 message requires a response of either a numeric or a filename. The message type table indicates for a type 1 message the number and location of responses in the response table. For a type 2 message, it indicates the location of a segment of code which reads and checks the validity of the response, and a pointer to the next message. The flow table contains, in a reentrant tree form, the entire logical flow for the statistical section. It contains an entry for every possible response for each type 1 message. Each entry contains an optional pointer to a segment of code to be executed and a pointer to the next message to be displayed. Finally, the response table contains the literal strings indicating all the valid responses for each type 1 message.

Each of the above tables is divided into two parts. The first part remains constant for each statistical section and references only those messages in the common message file. The Fortran code segments associated with these common messages also remains the same across sections. The second part of each table refers to the local message file associated with only that statistical section. Using this strategy, a bare minimum of modification needs to take place when new sections are created. The new messages particular to the new statistical sections need to be worded. The set of responses and the flows must be established. The rest of the work involves writing the several small segments of code to set up the binary coded parameter cards for the statistical programs in SOUPEX required by the particular statistical section.

Another feature existing in each statistical section is a series of special responses called interrupt words. They can be entered in place of either type of response and will cause the program flow to be temporarily interrupted while a specific function is performed. The previously mentioned replies 'HELP' and ' EXPERIENCED ', which alter the message level mode are interrupt words. The total set and their related functions are included in the IESS GUIDE appendix. One of my favorite interrupt features is the local message function. If a user types 'MESSAGE', the prompter will search his files for a file named IESSMESS. If the file exists, then its contents will be displayed. This allows teachers of different classes to have their own individual messages displayed to their students. The facility could also easily be expanded to have separate messages for each assignment or different individuals. Due to the table driven design of the statistical sections, modifications or additions of this type can be accomplished with very little effort.

Chapter 3 -- Correlations-Factor Analysis Statistical Section

```
"*************************************************************
"* THE FIRST SET OF VARIABLE DEFINITIONS REMAIN CONSTANT FOR ALL
"* STATISTICAL SECTIONS. THEY CONTAIN VARIABLES TO HOLD FILE NAMES,
"* ARRAYS FOR CARD IMAGES USED IN THE JOB STREAM, A SERIES OF
"* TEMPORARY VARIABLES, AND THE DEFINITIONS OF THE THREE MAIN
"* DRIVER TABLES.   FOLLOWING ARE DETAILED DESCRIPTIONS:
"* JOB      NAME OF FILE CONTAINING SOUPEX FILED OUTPUT (OS JOB #)
"* DAFILE   NAME OF FILE CONTAINING USERS DATA
"* FOFILE   NAME OF FILE CONTAINING USERS FORMAT
"* PRFILE   NAME OF USER FILE TO BE LISTED WITH 'FILE' COMMAND
"* LINE     HOLDS CARD IMAGES WHEN USER REVIEWS HIS DATA
"* REPLY    CONTAINS THE USERS REPLIES
"* HERE-
"* THERE    START AND END FLAGS FOR THE PRINT SUBROUTINE
"* TEMP8    TEMPORARY FILE NAME HOLDER
"* EXEC     CARD IMAGE OF THE 'EXEC SOUPEX' CARD
"* JOBLIB   JOBLIB CARD FOR RUNNING SOUPEX ON HASP
"* RBLANK   A BLANK FOR INITIALIZATION
"* C AND X  THE FLAGS USED AS PARAMETERS IN DISPLAYING MESSAGES TO USER
"* S1FILE   TEMPORARY FILE USED TO CONSTRUCT THE JOB STREAM
"* S2FILE   TEMPORARY FILE USED TO CONSTRUCT THE JOB STREAM
"* RESP     HOLDS ALL VALID RESPONSES FOR TYPE 1 MESSAGES
"* PR       SETS PRINT PARAMETER IN BINARY PARAMETER CARDS FOR SOUPEX
"* CARDS    SETS CARD INPUT UNIT NUMBER TO 5 FOR SOUPEX
"* IWAIT    LENGTH OF WAIT STATES ISSUED AFTER JOB SUBMISSION
"* CC       HOLDS CONDITION CODE AFTER SUBROUTINE CALLS
"* INEW     LOGICAL VARIABLE FOR NEW USER MODE
"* ELEVEL   CONTAINS CURRENT EXPERIENCE LEVEL
"* NOUT     PRINTER UNIT NUMBER
"* X-----   VARIABLES CONTAIN SUBROUTINE TYPE FOR CALL TO CCMES
"* MA - BA  USED IN EQUIVALENCES TO FORM PRINT FLAGS
"* MESS     CONTAINS FLAGS FOR SCANNING SOUPEX OUTPUT
"* CUES     EQUIVALENCED TO RESP ARRAY FOR LOOKING AT CHARACTERS
"* FLOW     FOR EVERY RESPONSE POINTS TO NEXT MESSAGE # AND CODE SEGMENT
"* IPLY-
"* IALF     EQUIVALENCED TO REPLY FOR LOOKING AT CHARACTERS AND INTEGERS
"* TRACE    CONTAINS THE PATH OF MESSAGES SO YOU CAN GO BACKWARDS
"* NTYPE    NUMBER OF MESSAGES FOR THIS SECTION
"* TYPE     DEFINES MESSAGE TYPES, # OF RESPONSES, AND FLOW INFORMATION
"* I        CONTAINS CURRENT MESSAGE NUMBER
"* IT       POINTER TO CURRENT ENTRY IN TRACE ARRAY
"*           ETC. FOR SEVERAL TEMPORARY VARIABLES.
"*************************************************************
REAL*8 JOB,DAFILE,FOFILE,PRFILE,LINE(10),REPLY,HERE,THERE,TEMP8
REAL*8 EXEC(10)/'//','EXEC','SOUPEX',7*' '/
REAL*8 JOBLIB(10)/'//JOBLIB',' DD DSN=','USER.POO','98.SOUPE',%
                  'X,DISP=S','HR',4*' '/
REAL*8 RBLANK/' '/,C/'KIMKIM'/,X/'KIMKIM'/
REAL*8 S1FILE/'DLGTENHU'/,S2FILE/'PJSWNCGL'/
REAL*8 RESP(95)
INTEGER PR/65536/,CARDS/5/,IWAIT/27/,CC,INEW/0/,ELEVEL,NOUT/6/
```

```
INTEGER XSEND/1/,XRUN/2/,XDEST/3/,XPRINT/4/,XOPEN/5/,XWAIT/6/
INTEGER MA(2)/'STA',0/,BA(2)/'END',0/,MESS(9),SM,EM
INTEGER*2 CUES(4,95),FLOW(2,95)
INTEGER*2 IPLY(4),IALF(8),TRACE(100),IT,NTYPE/50/,TYPE(4,50)
INTEGER*2 I,WHY,SURE,HALP,HA,HE,HI,HO,NTRY,A(4),B,D,E,U(4),V
EQUIVALENCE (REPLY,IPLY(1),IALF(1))
EQUIVALENCE (RESP(1),CUES(1,1))
EQUIVALENCE (A(1),C),(A(4),B),(U(1),X),(U(4),V)
EQUIVALENCE (MA(1),HERE),(MA(2),SM),(BA(1),THERE),(BA(2),EM)
COMMON IALF,HALP
"
"
"*************************************************************
"* THE FOLLOWING SET OF VARIABLE DEFINITIONS IS LOCAL TO THIS
"* PARTICULAR STATISTICAL SECTION. THEY CONTAIN VARIABLES
"* DEFINING THE MNEUMONICS OF THE SOUPEX PROGRAMS TO BE USED
"* AND ARRAYS TO BE FILLED IN DURING EXECUTION WITH BINARY
"* CODED PARAMETER INFORMATION FOR THE SOUPEX PROGRAMS.
"*************************************************************
REAL*8 PROGS(10),STONEY/'STONEY.'/,ENDS/'END.'/
REAL*8 COR/'COR.'/,COM/'COM.'/,PRI/'PRI.'/,VAR/'VAR.'/
INTEGER*4 CPARM(20),MPARM(20),PPARM(20),VPARM(20),XPARM(20),FORMAT(20)
INTEGER*4 DATA/'DATA'/,END(20)/'END#',19*' '/
"
"
"*************************************************************
"* THE TYPE ARRAY CONTAINS AN ENTRY FOR EVERY MESSAGE. THE FIRST
"* 30 MESSAGES ARE COMMON TO ALL SECTIONS AND FROM 31 ON ARE
"* LOCAL TO THIS STATISTICAL SECTION. THERE ARE TWO CLASSES
"* OF MESSAGES. TYPE 1 MESSAGES REQUIRED A RESPONSE CHOSEN
"* FROM A SET OF RESPONSES. THE ENTRY IN THIS TABLE WOULD APPEAR AS:
"*      MESSAGE #,# RESPONSES,0,#CHARS. IN RESPONSE.
"* TYPE 2 MESSAGES REQUIRING A NUMERICAL RESPONSE APPEAR AS:
"*      MESSAGE #,0,CODE SEGMENT #,# OF NEXT MESSAGE.
"*
"* MESSAGE # IS A NUMBER FROM 1 TO 60 IDENTIFYING THE MESSAGE.
"* # RESPONSES IS THE SIZE OF THE SET OF RESPONSES IN THE RESP ARRAY.
"* #CHARS. IN RESPONSE IS MINIMUM # OF CHARS. TO IDENTIFY RESPONSE.
"* CODE SEGMENT # POINTS TO CODE WHICH READS AND VERIFIES NUMBER.
"* # OF NEXT MESSAGE INDICATES FLOW AFTER NUMERIC IS PROCESSED.
"*
"* THE CODE SEGMENTS AND FLOWS FOR TYPE 1 MESSAGES ARE PAIRED
"* WITH THE SET OF RESPONSES IN THE RESP ARRAY.
"*************************************************************
DATA TYPE/%
" EMPTY
 1, 4, 0, 3,%
"               SAME,DIF,TYPE,CREATE.
 2, 4, 0, 1,%
"               FORMAT FILE.
 3, 0,13,22,%
"               DATA FILE.
 4, 0,11, 3,%
```

```
"                FORMAT-DATA FILE.
 5, 0,12,22,%
"                (#)(FORMAT).
 6, 0,14,22,%
"                SEE FEW ROWS.
 7, 2, 0, 1,%
"                CON. WAIT OR RETURN.
 8, 2, 0, 1,%
"                ENTIRE OUT. OR SEC.
 9, 2, 0, 1,%
"                PRINTED COPY.
10, 2, 0, 1,%
"                EXPRESS OR HASP.
11, 2, 0, 1,%
"                WHICH TERMINAL.
12,13, 0, 3,%
"                KEEP OR DEST.
13, 2, 0, 1,%
"                JOB #.
14, 0,23, 9,%
"                SAME ANAL. DIF. DATA.
15, 2, 0, 1,%
"                STOP OR CONT.
16, 2, 0, 1,%
"                SET,DEF,REV,BRA.
17, 4, 0, 1,%
"                # VAR.
18, 0,30,19,%
"                MAX. DIGITS.
19, 0,31,20,%
"                FILE NAME.
20, 0,32,21,%
"                CREATE DATA.
21, 0,25,22,%
"                RUN OR START OVER.
22, 3, 0, 1,%
"                SEE MORE DATA.
23, 2, 0, 1,%
24, 0, 0, 0,%
25, 0, 0, 0,%
26, 0, 0, 0,%
27, 0, 0, 0,%
28, 0, 0, 0,%
29, 0, 0, 0,%
30, 0, 0, 0,%
31, 2, 0, 1,%
32, 3, 0, 1,%
33, 3, 0, 1,%
34, 3, 0, 1,%
35, 0, 0, 0,%
36, 0, 0, 0,%
37, 0, 0, 0,%
38, 2, 0, 1,%
```

```
39, 3, 0, 3,%
40, 2, 0, 1,%
41, 4, 0, 1,%
42, 0,63,43,%
43, 0,64,44,%
44, 2, 0, 1,%
45, 2, 0, 1,%
46, 2, 0, 1,%
47, 2, 0, 1,%
48,13, 0, 3,%
49, 3, 0, 1,%
50, 3, 0, 1/
"
"
"*************************************************************
"* THE RESP ARRAY CONTAINS LITERALS IDENTIFYING ALL THE VALID
"* RESPONSES FOR THE INDIVIDUAL MESSAGES. TO MATCH UP A SET OF
"* RESPONSES WITH THE CORRECT MESSAGE USE THE LINE NUMBERS.
"* FOR EXAMPLE, THE ENTRY FOR MESSAGE 7 IS ON LINE 307 IN THE
"* TYPE ARRAY, LINE 407 IN THE RESP ARRAY, AND LINE 507 IN THE
"* FLOW ARRAY. THE TYPE 2 MESSAGES IN THE TYPE ARRAY DO NOT HAVE
"* A LIST OF RESPONSES, BUT HAVE A LINE NUMBER PLACE HOLDER IN THE
"* FORM OF A COMMENT INDICATING THE FUNCTION OF THE MESSAGE.
"*************************************************************
DATA RESP/%
'F A C','R E G','F R E','L I N',%
'S','D','T','C',%
"FORMAT FILE
"DATA FILE
"FORMAT-DATA FILE
"(#)(FORMAT)
'Y','N',%
'C O N','R E T',%
'E N T','S E C',%
'Y E S','N O',%
'E X P','H A S',%
'C O M','L O C','P L O','P S Y','A G R','C I V','C H E',%
'O I R','S G S','S W S','U I C','S S U','R M 1 1',%
'K','D',%
"JOB#
'Y','N',%
'S','C',%
'S','D','R','B',%
"# VAR.
"MAX. DIGITS
"FILE NAME
"CREATE DATA
'E','H','B',%
'Y E S','N O',%
'Y','N',%
'Y','N','S',%
'Y','N','S',%
'Y','N','S',%
```

```
"EMPTY
"EMPTY
"EMPTY
'N O','Y E S',%
'C O R','C O V','C R O',%
'Y E S','N O',%
'L','M','U','S',%
"MAX. # FACTORS
"PERCENT VARIANCE
'Y E S','N O',%
'Y E S','N O',%
'N O R','R A W',%
'Y E S','N O',%
'M E A','C O R','C O V','C R O','C O M','P R I','V A R',%
'D O N','F I N','N O N','E N D','N O','E N T',%
'1','2','3',%
'S','D','R'/
"
"
"*************************************************************
"* THE FLOW ARRAY CONTAINS AN ENTRY FOR EVERY RESPONSE IN THE RESP
"* ARRAY. EACH ENTRY CONTAINS 2 PARTS. THE FIRST PART INDICATES THE
"* NUMBER OF A CODE SEGMENT TO BE EXECUTED IF THAT RESPONSE IS GIVEN.
"* THE SECOND PART INDICATES THE # OF THE NEXT MESSAGE TO BE DISPLAYED.
"* IF THE CODE SEGMENT # IS ZERO, THEN NO CODE IS EXECUTED AND
"* THE NEXT MESSAGE IS DISPLAYED IMMEDIATELY. AS WITH THE RESPZZZZZZZZZZ
"* ARRAY, TYPE 2 MESSAGES HAVE NO ENTRIES IN THIS TABLE. A COMMENT
"* CARD IS INCLUDED AT THERE LINE NUMBER IN THE ARRAY. THE FLOW
"* INFORMATION FOR TYPE 2 MESSAGES IS CONTAINED IN THE TYPE ARRAY.
"*************************************************************
DATA FLOW/%
20,00,  20,00,  20,00,  20,00,%
24,05,  24,04,  24,04,  00,18,%
"FORMAT FILE
"DATA FILE
"FORMAT-DATA FILE
"(#)(FORMAT)
27,23,  00,06,%
15,09,  16,15,%
17,48,  00,48,%
00,11,  00,13,%
21,12,  21,12,%
22,13,  22,13,  22,13,  22,13,  22,13,  22,13,  22,13,%
22,13,  22,13,  22,13,  22,13,  22,13, 22,13,%
36,15,  35,15,%
"JOB #
00,02,  00,16,%
33,00,  00,17,%
26,31,  26,49,  26,14,  34,26,%
"# VAR.
"MAX. DIGITS
"FILE NAME
"CREATE DATA
```

```
58, 9,  58, 9, 00,16,%
28,23,  29,06,%
"   NONE
"   NONE
"   NONE
"   NONE
"   NONE
"   NONE
"   NONE
51,32,  00,32,%
52,33,  00,33,  00,02,%
53,34,  00,34,  00,02,%
54,02,  00,02,  00,02,%
"EMPTY
"EMPTY
"EMPTY
00,22,  00,39,%
59,40,  59,42,  59,42,%
00,41,  00,42,%
62,42,  62,42,  62,42,  62,42,%
"MAX. # FACTORS
"PERCENT VARIANCE
65,45,  00,45,%
66,46,  00,22,%
00,22,  67,22,%
68,22,  00,22,%
69,48,  69,48,  69,48,  69,48,  69,48,  69,48,  69,48,%
00,10,  00,10,  00,10,  00,10,  00,10,  17,48,%
78,02,  79,02,  80,02,%
26,31,  26,49,  26,14/
"
"
"*****************************************************************
"* THE MESS ARRAY CONTAINS THE FLAGS TO BE USED IN SCANNING THE
"* SECTIONS OF THE OUTPUT.
"*****************************************************************
DATA MESS/%
'MEAN','CORR','COVA','CROS','COMP','PRIP','VARP'/
"*
"*
"*****************************************************************
"* THE FOLLOWING DO LOOP REPLACES THE FIRST COLUMN OF ENTRIES
"* IN THE TYPE ARRAY WITH POINTERS TO EACH MESSAGE'S RESPONSES IN
"* THE RESP ARRAY. THIS IS DONE BY ACCUMULATING THE SUM OF THE SECOND
"* COLUMN OF THE TYPE ARRAY, WHICH CONTAINS THE # OF RESPONSES
"* FOR EACH MESSAGE.
"*****************************************************************
TYPE(1,1)=1
DO 4 I=2,NTYPE
4 TYPE(1,I)=TYPE(1,I-1)+TYPE(2,I-1)
"                              INITIAL MESSAGE MODE IS FAMILIAR.
5 ELEVEL=2
"                              START WITH MESSAGE # 50
```

```
I=50
IT=1
TRACE(1)=0
10 IF(INEW.EQ.1)GOTO 11
IF(ELEVEL.EQ.1)ELEVEL=2
11 IF(I)5,5,12
12 IJ=I
IF(I.GT.30)IJ=I-30
"*************************************************************
"* THE FOLLOWING 8 LINES OF CODE CALCULATE THE STARTING AND ENDING
"* FLAGS FOR THE MESSAGE TO BE DISPLAYED. THE VARIABLE 'I' CONTAINS
"* THE MESSAGE NUMBER AND 'ELEVEL' CONTAINS THE MESSAGE MODE.
"* THE NUMBER CALCULATED IS STUFFED INTO ALPHABETIC VARIABLES THROUGH
"* THE USE OF EQUIVALENCE STATEMENTS. THESE VARIABLES ARE SUPPLIED
"* AS PARAMETERS TO THE PRINT SUBROUTINE BELOW.
"*************************************************************
N=IJ*3+ELEVEL-3
NN=N+1
E=MOD(N,10)
D=N/10
B=E+D*256+61680
E=MOD(NN,10)
D=NN/10
V=E+D*256+61680
20 IF(I-30)22,22,24
"                              DISPLAY A MESSAGE FROM THE COMMON FILE
22 CALL PRINT('1000.L.I',C,X,CC,'E')
GOTO 26
"                              DISPLAY A MESSAGE FROM THE LOCAL FILE.
24 CALL PRINT('1000.L.F',C,X,CC,'E')
26 NTRY=1
GOTO 40
30 IF(ELEVEL.GT.1)GOTO 35
ELEVEL=1
32 I=TRACE(IT)
"                              GO BACKWARDS TO THE LAST MESSAGE.
IT=IT-1
GOTO 10
35 ELEVEL=ELEVEL-1
"                              DROP TO A LOWER MESSAGE MODE.
GOTO 11
40 IF(TYPE(2,I).GT.0)GOTO 44
"                              REQUIRES A NUMERIC OR FILENAME RESPONSE
IT=IT+1
"                              RECORD OLD MESSAGE # IN TRACE TABLE.
TRACE(IT)=I
"                              FIND CODE SEGMENT # TO BE EXECUTED.
HE=TYPE(3,I)
"                              RECORD # OF NEXT MESSAGE.
I=TYPE(4,I)
GOTO 55
"                              READ REPLY.
44 READ(5,3004)IPLY
```

```
"                                CHECK FOR INTERRUPT WORD.
CALL STANR(&94)
"                                WHERE DO RESPONSES START IN RESP ARRAY?
HO=TYPE(1,I)
"                                WHERE DO RESPONSES END IN RESP ARRAY?
HA=HO+TYPE(2,I)-1
"                                SET # OF CHARS. REQUIRED IN RESPONSE.
LSCAN=TYPE(4,I)
"                                FOLLOWING IS LOOP TO SCAN RESPONSE.
IRES=0
DO 58 SURE=HO,HA
IRES=IRES+1
DO 52 WHY=1,LSCAN
IF(IPLY(WHY).NE.CUES(WHY,SURE))GOTO 58
52 CONTINUE
"                                FOUND A VALID RESPONSE.
IT=IT+1
"                                SAVE OLD MESSAGE # IN TRACE ARRAY.
TRACE(IT)=I
"                                RECORD # OF NEXT MESSAGE.
I=FLOW(2,SURE)
"                                FIND CODE SEGMENT # TO BE EXECUTED.
HE=FLOW(1,SURE)
55 IF(HE.EQ.0)GOTO 10
IF(HE-50)56,56,57
"                                CODE SEGMENT BRANCH FOR COMMON MESSAGES.
56 HALP=HE-10
GOTO(1110,1120,1130,1140,1150,1160,1170,1180,1190,1200,%
     1210,1220,1230,1240,1250,1260,1270,1280,1290,1300,%
     1310,1320,1330,1340,1350,1360,1370,1380,1390,1400),HALP
CALL ERR(10021)
"                                CODE SEGMENT BRANCH FOR LOCAL MESSAGES.
57 HALP=HE-50
GOTO(2010,2020,2030,2040,2050,2060,2070,2080,2090,2100,%
     2110,2120,2130,2140,2150,2160,2170,2180,2190,2200,%
     2210,2220,2230,2240,2250,2260,2270,2280,2290,2300),HALP
CALL ERR(10023)
58 CONTINUE
"                                RESPONSE IS NOT VALID.
60 IF(NTRY.GE.3)GOTO 30
"                                WRITE 'TRY AGAIN' MESSAGE.
WRITE(NOUT,62)
62 FORMAT(' TRY AGAIN ')
"                                RETURN TO ABOVE CODE TO READ RESPONSE.
NTRY=NTRY+1
GOTO 40
68 I=TRACE(IT)
"                                BAD NUMERIC OR FILENAME ERROR RETURN
IT=IT-1
GOTO 60
70 WRITE(NOUT,72)
"                                NOT ENOUGH PLORTS SPACE ERROR RETURN
72 FORMAT(' YOUR AVAILABLE SPACE MUST BE INCREASED TO CONTINUE.')
```

```
GOTO 200
90 I=TRACE(IT)
IT=IT-1
"                                BRANCH GOTO FOR INTERRUPT WORDS
94 GOTO(5,110,32,30,30,130,160,190,%
     200,300,400,500,600,650,700,900,%
     900,900,900,900,900,900,900,900),HALP
CALL ERR(10027)
110 I=1
"                                BRANCH TO STATISTICAL SECTION.
GOTO 10
130 ELEVEL=3
"                                EXPERIENCED MESSAGE MODE
INEW=0
GOTO 10
160 ELEVEL=2
"                                FAMILIAR MESSAGE MODE
INEW=0
GOTO 10
190 ELEVEL=1
"                                NEW USER MESSAGE MODE
INEW=1
GOTO 10
200 WRITE(NOUT,210)IT
"                                STOP COMMAND
210 FORMAT(' STOPPED AFTER EXECUTING',I3,' STEPS.')
STOP
300 WRITE(NOUT,*)IT
"                                STEP COMMAND
GOTO 10
400 IF(JOB.NE.0.)WRITE(NOUT,420)JOB
"                                DISPLAY JOB NAME POINTER.
420 FORMAT(' JOB# IS',A9)
GOTO 10
500 WRITE(NOUT,501)
501 FORMAT(' FILENAME TO BE LISTED.')
READ(5,502)PRFILE
"                                LIST A USER SUPPLIED FILE.
502 FORMAT(A8)
CALL PRINT(PRFILE,'FIRST','LAST',CC)
CALL CCMES(XPRINT,CC)
GOTO 10
600 IPATH=3
"                                REVIEW FILED OUTPUT
I=14
GOTO 10
650 WRITE(6,660)
660 FORMAT(' TYPE JOB#')
"                                CHANGE CONTENTS OF JOB NAME POINTER.
TEMP8=JOB
READ(5,502)JOB
CALL WAIT(JOB,1,CC)
IF(CC.EQ.0)GOTO 10
```

```
WRITE(NOUT,1235)JOB
JOB=TEMP8
GOTO 10
700 CALL PRINT('IESSMESS','FIRST','LAST',CC)
IF(CC.EQ.0)GOTO 10
"                               LIST USER IESMESS FILE.
WRITE(NOUT,730)
730 FORMAT(' NO MESSAGE FILE EXISTS. MESSAGE OPTION IGNORED.')
GOTO 10
900 WRITE(NOUT,910)
"                               PROFANE RESPONSE RETURN
910 FORMAT(' PROFANE REMARKS WILL NOT HELP THE SITUATION½'/%
' TYPE ''HELP'' INSTEAD. ')
GOTO 10
"
"
"*****************************************************************
"* FOLLOWING ARE THE CODE SEGMENTS WHICH ARE COMMON TO ALL
"* STATISTICAL SECTIONS. THEY ARE BRANCHED TO THROUGH THE
"* COMPUTED GOTO STATEMENT ABOVE. AFTER EXECUTION, MOST OF THE
"* CODE SEGMENTS BRANCH TO STATEMENT NUMBER 10 WHERE THE NEXT
"* MESSAGE IS DISPLAYED.
"*****************************************************************
1110 READ(5,3411)IPLY,DAFILE
"                               READ DATA FILE NAME.
CALL STANR(&90)
CALL EXIST(DAFILE,&4444,&68)
IF(IDATA.EQ.3)I=7
GOTO 10
1120 READ(5,3411)IPLY,DAFILE
"                               READ FORMAT-DATA FILE NAME.
CALL STANR(&90)
CALL EXIST(DAFILE,&4444,&68)
IFRONT=1
IFILE=0
GOTO 1144
1126 CALL ERR(10031)
1130 READ(5,3411)IPLY,FOFILE
"                               READ FORMAT FILE NAME.
CALL STANR(&90)
CALL EXIST(FOFILE,&4444,&68)
IFILE=1
IFRONT=0
GOTO 1144
1140 READ(5,3019)IPLY,(FORMAT(HALP),HALP=2,20)
"                               READ USER SUPPLIED FORMAT.
CALL STANR(&90)
IFILE=0
IFRONT=0
1144 IF(IPATH.EQ.1)I=38
GOTO 10
1150 CALL WAIT(JOB,IWAIT,CC)
"                               ENTER WAIT STATE FOR SOUPEX OUTPUT.
```

```
CALL CCMES(XWAIT,CC)
IPATH=4
IF(CC)70,10,1156
1156 I=8
GOTO 10
1160 WRITE(NOUT,1161)JOB
"                                DISPLAY SOUPEX JOB NAME.
1161 FORMAT(' YOUR JOB # IS ',A8/%
' YOU WILL NEED TO KNOW THIS # WHEN YOU RETURN TO REVIEW THE OUTPUT.')
GOTO 10
1170 CALL LPRINT(JOB,'STA STA','END END',CC)
"                                LIST ENTIRE SOUPEX OUTPUT.
CALL CCMES(XPRINT,CC)
IF(CC)4120,10,4160
1180 GOTO 1800
1190 GOTO 1800
1200 CALL BRANCH(IRES)
"                                BRANCH TO ANOTHER STATISTICAL SECTION.
CALL ERR(10035)
1210 ISYS=IRES
"                                PRINTED COPY TO HASP OR EXPRESS?
GOTO 10
1220 IW=IRES
"                                PRINT COPY AT WHICH TERMINAL?
1225 CALL OUTPUT(JOB,ISYS,IW)
GOTO 10
1230 READ(5,3411)IPLY,JOB
"                                READ NAME OF FILED OUTPUT FOR REVIEW.
CALL STANR(&90)
CALL WAIT(JOB,1,CC)
IF(CC.EQ.0)GOTO 10
WRITE(NOUT,1235)JOB
1235 FORMAT(' UNABLE TO FIND FILED OUTPUT CALLED ',A8)
GOTO 68
1240 IDATA=IRES
"                                SET FORM OF DATA FLAG.
GOTO 10
1250 CALL OPEN(3,DAFILE,'OUTPUT',CC)
"                                CREATE THE DATA AT THE TERMINAL.
CALL CCMES(XOPEN,CC)
IF(CC.EQ.1)GOTO 70
CALL CREATE(IVAR,IMAX,&1258)
IFRONT=1
IFILE=0
CALL CLOSE(3)
GOTO 1144
1258 CALL CLOSE(3)
CALL DEST(DAFILE,PP)
GOTO 90
1260 IPATH=IRES
"                                INITIALIZATION SECTION.
IFILE=0
IFRONT=0
```

```
DO 1261 HALP=1,20
CPARM(HALP)=0
MPARM(HALP)=0
PPARM(HALP)=0
VPARM(HALP)=0
FORMAT(HALP)=0
1261 XPARM(HALP)=0
FORMAT(1)=DATA
DO 1262 HALP=1,10
1262 PROGS(HALP)=RBLANK
TRACE(2)=TRACE(IT)
IT=2
PROGS(2)=COR
CPARM(1)=5
DAFILE=0.
FOFILE=0.
JOB=0.
PROGS(1)=STONEY
PROGS(6)=ENDS
GOTO 10
1270 CALL OPEN(3,DAFILE,'INPUT',CC)
"                                LOOK AT USER'S DATA FILE.
INUMB=3
1272 DO 1274 HALP=1,INUMB
READ(3,3010,END=1276)LINE
1274 WRITE(NOUT,3110)LINE
GOTO 10
1276 WRITE(NOUT,1277)
1277 FORMAT(' END OF FILE--WILL START OVER IF MORE REQUESTED.')
CALL CLOSE(3)
CALL OPEN(3,DAFILE,'INPUT',CC)
GOTO 10
1280 INUMB=INUMB+INUMB+INUMB
"                                LOOK AT MORE ROWS OF DATA FILE.
GOTO 1272
1290 CALL CLOSE(3)
"                                CLOSE USER'S DATA FILE.
GOTO 10
1300 READ(5,3418)IPLY,IALF
"                                READ # OF VARIABLES TO BE CREATED.
CALL STANR(&90)
CALL ALFTOI(IALF,IVAR,&68)
IF(IVAR.GT.30)GOTO 68
IF(IVAR.LT.1)GOTO 68
GOTO 10
1310 READ(5,3418)IPLY,IALF
"                                READ MAX. # DIGITS PER VARIABLE.
CALL STANR(&90)
CALL ALFTOI(IALF,IMAX,&68)
IF(IMAX.GT.17)GOTO 68
IF(IMAX.LT.1)GOTO 68
IF(IVAR*IMAX+IVAR-73)10,10,1318
1318 WRITE(NOUT,1319)
```

```
1319 FORMAT(' # VAR. AND/OR MAX. DIGITS TOO LARGE.')
I=24
GOTO 10
1320 READ(5,3411)IPLY,DAFILE
"                              READ FILENAME TO PUT CREATED DATA IN.
CALL STANR(&90)
CALL OPEN(3,DAFILE,'INPUT',CC)
IF(CC.EQ.2)GOTO 10
CALL CLOSE(3)
WRITE(NOUT,1328)
1328 FORMAT(' FILE ALREADY EXISTS.')
GOTO 68
1330 GOTO 200
"                              BRANCH TO 'STOP' ROUTINE.
1340 GOTO 110
"                              BRANCH TO ANOTHER STATISTICAL SECTION.
1350 CALL DEST(JOB,PP)
"                              DESTROY USER'S FILED OUTPUT.
1360 IF(IPATH.EQ.3)I=16
"                              SKIP NEXT MESSAGE IF ON REVIEW PATH.
GOTO 10
1370 CONTINUE
1380 CONTINUE
1390 CONTINUE
1400 CONTINUE
1800 CALL ERR(10017)
"
"
"*************************************************************
"* FOLLOWING ARE THE CODE SEGMENTS WHICH APPEAR ONLY IN THIS
"* STATISTICAL SECTION. MOST OF THEM SERVE THE FUNCTION OF
"* FILLING IN THE BINARY CODED PARAMETER CARDS SUPPLIED TO
"* THE SOUPEX PROGRAMS. THE JOB STREAM FOR THE PARTICULAR
"* ANALYSES TO BE PERFORMED IS ALSO BUILT IN THIS SECTION.
"*************************************************************
2010 CPARM(2)=PR
"                              PRINT MEANS AND STANDARD DEVAITIONS
IDATA=0
GOTO 10
2020 CPARM(3)=PR
"                              PRINT CORRELATIONS MATRIX.
GOTO 10
2030 CPARM(5)=PR
"                              PRINT CROSS-PRODUCTS MATRIX.
GOTO 10
2040 CPARM(4)=PR
"                              PRINT COVARIANCE MATRIX.
GOTO 10
2050 GOTO 2800
2060 GOTO 2800
2070 GOTO 2800
2080 CALL OPEN(1,S1FILE,'OUTPUT',CC)
IF(CC.EQ.1)GOTO 70
```

```
IF(CC.GT.1)CALL ERR(10007)
IF(IRES.EQ.2)WRITE(1,3010)JOBLIB
"                                SET UP JOB STREAM FOR SOUPEX RUN.
WRITE(1,3010)EXEC
WRITE(1,3010)PROGS
WRITE(1,3220)CPARM,XPARM
IF(IFRONT.NE.0)GOTO 2082
IF(IFILE.NE.0)GOTO 2082
WRITE(1,3020)FORMAT
2082 CALL CLOSE(1)
CALL SEND(S1FILE,'U',CC)
IF(CC.NE.0)CALL ERR(10009)
IF(IFILE.NE.0)CALL SEND(FOFILE,'U',CC)
IF(CC.NE.0)GOTO 4800
CALL SEND(DAFILE,'U',CC)
IF(CC.NE.0)GOTO 4800
CALL OPEN(2,S2FILE,'OUTPUT',CC)
IF(CC.EQ.1)GOTO 70
IF(CC.GT.1)CALL ERR(10011)
WRITE(2,3020)END
IF(PROGS(3).NE.RBLANK)WRITE(2,3220)MPARM,XPARM
IF(PROGS(4).NE.RBLANK)WRITE(2,3220)PPARM,XPARM
IF(PROGS(5).NE.RBLANK)WRITE(2,3220)VPARM,XPARM
CALL CLOSE(2)
CALL SEND(S2FILE,'U',CC)
IF(CC.NE.0)CALL ERR(10013)
JOB=0.
IF(IRES.EQ.1)CALL XPRESS(JOB,CC)
IF(IRES.EQ.2)CALL HASP(JOB,CC)
CALL CCMES(XRUN,CC)
CALL DEST(S1FILE,PP)
CALL DEST(S2FILE,PP)
IF(CC)4200,2088,4300
2088 WRITE(NOUT,2089)
2089 FORMAT(' YOUR JOB HAS BEEN SUBMITTED FOR EXECUTION.'/%
' WE ARE NOW WAITING FOR THE OUTPUT.')
GOTO 1150
2090 CPARM(IRES+2)=PR+11
"                                FACTOR WHICH MATRIX?
IRON=IRES-1
GOTO 10
2100 GOTO 2800
2110 GOTO 2800
2120 PROGS(3)=COM
"                                SET COMMUNALITIES PARAMETERS.
IIN=IRES
IRON=IIN
IF(IIN.EQ.3)GOTO 68
MPARM(1)=11
MPARM(2)=11
MPARM(3)=IIN
GOTO 10
2130 PROGS(4)=PRI
```

```
"                                   SET PRINCIPAL AXIS PARAMETERS
PPARM(1)=11
PPARM(2)=PR+12
READ(5,3418)IPLY,IALF
CALL STANR(&90)
CALL ALFTOI(IALF,IIN,&68)
IF(IIN.GT.30)GOTO 68
PPARM(3)=IIN
GOTO 10
2140 READ(5,3418)IPLY,IALF
"                                   PERCENT VARIANCE TO EXTRACT.
CALL STANR(&90)
CALL ALFTOI(IALF,IIN,&68)
IF(IIN.GT.100)GOTO 68
PPARM(4)=IIN
GOTO 10
2150 PPARM(5)=1
"                                   STOP FACTOR AT UNITY.
GOTO 10
2160 PROGS(5)=VAR
"                                   SET VARIMAX PARAMETERS.
VPARM(1)=12
VPARM(2)=PR
IF(IRON.EQ.0)GOTO 2166
VPARM(3)=1
GOTO 10
2166 I=22
GOTO 10
2170 VPARM(4)=1
GOTO 10
2180 VPARM(5)=PR
GOTO 10
2190 SM=MESS(IRES)
"                                   DISPLAY WHICH SECTION OF OUTPUT.
EM=SM
GOTO 4100
2200 GOTO 2800
2210 GOTO 2800
2220 GOTO 2800
2230 GOTO 2800
2240 GOTO 2800
2250 GOTO 2800
2260 GOTO 2800
2270 GOTO 2800
2280 DO 2281 HALP=2,7
"                                   DEFAULT PARAMETER PATH 1.
2281 CPARM(HALP)=PR
GOTO 10
2290 CPARM(2)=PR
"                                   DEFAULT PARAMETER PATH 2.
CPARM(3)=PR+11
PROGS(4)=PRI
PPARM(1)=11
```

```
PPARM(2)=12
PPARM(3)=0
PPARM(4)=100
PROGS(5)=VAR
VPARM(1)=12
VPARM(2)=PR
GOTO 10
2300 GOTO 68
2800 CALL ERR(10017)
"*****************************************************************
"* FORMATS ARE STORED IN SAME AREA TO AVOID DUPLICATION.
"*****************************************************************
3004 FORMAT(4A1)
3010 FORMAT(10A8)
3019 FORMAT(4A1,T1,19A4)
3020 FORMAT(20A4)
3110 FORMAT(1X,9A8/1X,A8)
3220 FORMAT(20A4/20A4)
3411 FORMAT(4A1,T1,A8)
3418 FORMAT(4A1,T1,8A1)
"*****************************************************************
"* ERROR MESSAGES AND ERROR RETURNS FROM THE VARIOUS FILE
"* HANDLING AND JOB ENTRY SUBROUTINES ARE GROUPED TOGETHER.
"*****************************************************************
4100 EM=SM
CALL PRINT(JOB,HERE,THERE,CC)
CALL CCMES(XPRINT,CC)
IF(CC)4120,10,4160
4120 WRITE(NOUT,4125)
4125 FORMAT(' SECTION WAS NOT CALCULATED OR ERROR OCCURED.'/%
' LIST ENTIRE OUTPUT TO DETERMINE.')
GOTO 10
4160 CALL ERR(10029)
4200 WRITE(NOUT,4202)
4202 FORMAT(' I.E.S.S. WILL NOT RUN WHEN RUN IS DISABLED.'/%
' PLEASE TRY AGAIN LATER.')
GOTO 200
4300 CALL ERR(10033)
4444 CALL ERR(10015)
4800 WRITE(NOUT,4802)
4802 FORMAT(' ERROR IN DATA AND/OR FORMAT FILES.'/%
'CORRECT DATA FILES AND TRY AGAIN.')
I=16
GOTO 10
END
"
"
"*****************************************************************
"* THE CREATE SUBROUTINE IS USED TO CREATE A FILE CONSISTING
"* OF A FORMAT CARD FOLLOWED BY OBSERVATIONS OF DATA. IT IS
"* FOR THE USER WHO WANTS TO TYPE HIS DATA IN AT THE TERMINAL
"* AND DOES WANT TO HAVE TO CREATE PLORTS CARD IMAGE DATA
"* FILES BEFORE HE USES I.E.S.S.
```

```
"*
"* VARIABLES ARE ENTERED ONE AT A TIME SEPARATED BY BLANKS.
"* WHEN THE ENTIRE OBSERVATION IS TYPED, THEN 'CR' IS HIT.
"* THE # OF VARIABLES AND  MAX. # OF DIGITS PER VARIABLE ARE PASSED
"* AS PARAMETERS FROM THE MAIN PROGRAM. USING THIS INFORMATION
"* AND SCANNING THE INPUT CAREFULLY, THE SUBROUTINE REJECTS EVERYTHING
"* EXCEPT VALID OBSERVATIONS.  WHEN THE USER IS THROUGH ENTERING
"* HIS DATA, HE TYPES 'EOF'.  HIS CREATED DATA FILE IS CLOSED AND
"* THE SUBROUTINE RETURNS TO THE MAIN PROGRAM FOR THE NEXT MESSAGE.
"*****************************************************************
SUBROUTINE CREATE(IVAR,IMAX,*)
REAL*8 REPLY,EOF/'E O F'/,PLACE
INTEGER IVAR,IMAX,ICOL,OCOL,NDEC,NOUT/6/,N3/3/,IROW
INTEGER*2 ILINE(72),OLINE(80),BLANK/' '/,BUF(74),IPLY(4),HALP
INTEGER*2 CHAR(12)/'0','1','2','3','4','5','6','7','8','9','-','.'/
INTEGER*2 D(10)/'DA','TA','(',0,')(',0,'F ',0,'.0',')'/
EQUIVALENCE (REPLY,IPLY(1))
COMMON REPLY,PLACE,HALP
IROW=0
D(4)=IVAR
D(6)=IVAR
D(8)=IMAX+1
WRITE(N3,3)D
3 FORMAT(2A2,A1,I2,A2,I2,A1,I2,A2,A1)
4 READ(5,5)IPLY,ILINE
5 FORMAT(4A1,T1,72A1)
CALL STANR(&98)
IF(REPLY.EQ.EOF)GOTO 99
ICOL=0
OCOL=0
DO 60 I=1,IVAR
NDEC=0
J=0
10 J=J+1
12 ICOL=ICOL+1
IF(ICOL.GT.72)GOTO 90
IF(ILINE(ICOL).EQ.BLANK)GOTO 40
DO 20 M=1,12
IF(ILINE(ICOL).EQ.CHAR(M))GOTO 30
20 CONTINUE
GOTO 90
30 IF(M.NE.11)GOTO 36
IF(J.NE.1)GOTO 90
36 IF(M.EQ.12)NDEC=NDEC+1
BUF(J)=ILINE(ICOL)
GOTO 10
40 IF(J.EQ.1)GOTO 12
IF(NDEC.GT.1)GOTO 90
JJ=J-1
IF(JJ.GT.IMAX)GOTO 90
DO 46 M=JJ,IMAX
OCOL=OCOL+1
46 OLINE(OCOL)=BLANK
```

```
DO 52 M=1,JJ
OCOL=OCOL+1
52 OLINE(OCOL)=BUF(M)
60 CONTINUE
JJ=OCOL+1
DO 62 M=JJ,80
62 OLINE(M)=BLANK
WRITE(N3,66)OLINE
66 FORMAT(80A1)
WRITE(NOUT,70)
70 FORMAT(' NEXT LINE OR ''EOF''.'/)
IROW=IROW+1
GOTO 4
90 WRITE(NOUT,92)
92 FORMAT(' BAD LINE. TRY AGAIN.'/)
GOTO 4
98 RETURN 1
99 HALP=5
IF(IROW.LT.1)RETURN 1
WRITE(NOUT,100)IROW
100 FORMAT(1X,I4,' ROWS OF DATA HAVE BEEN CREATED.')
RETURN
END
"
"
"*************************************************************
"* OUTPUT IS USED TO CREATE THE JOB WHICH PROVIDES HARD PRINTED
"* COPY OF THE FILED OUTPUT WHICH THE USED HAS BEEN VIEWING.
"* SUPPLIED AS PARAMETERS TO THE ROUTINE ARE THE NAME OF THE
"* FILE WHICH CONTAINS THE FILED SOUPEX OUTPUT, THE SYSTEM TO SUBMIT
"* THE JOB TO (HASP OR EXPRESS), AND THE TERMINAL AT WHICH THE
"* COPY SHOULD BE PRINTED. TO MAKE THE COPY MORE READABLE,
"* THE PRINT FLAGS CREATED IN THE OUTPUT BY SOUPEX ARE ELIMINATED.
"*************************************************************
SUBROUTINE OUTPUT(NAME,ISYS,IWHERE)
REAL*8 NAME,FROM/'STA STA '/,TO/'END END'/,COMP,LINE(9)
REAL*8 IDB/'/*ID '/,PRE/'PRINT='/,PLACE(13)
REAL*8 E1/'// EXEC '/,E2/' CARDS '/,LIST/'  LIST  '/
REAL*8 JOB,ONAME/'INTPEXLI'/
REAL*4 SS/'STA '/,EE/'END '/,CO
INTEGER NOUT/6/
EQUIVALENCE (LINE(1),COMP,CO)
DATA PLACE/'COMM','LOCAL','PLORTS','PSYCH','AGRIC','CIVIL','CHEM',%
          'OIR','SGS','SWS','UIC','SSU','RM11'/
CALL OPEN(8,NAME,'INPUT',IC)
IF(IC.GT.0)GOTO 300
CALL OPEN(9,ONAME,'OUTPUT',IC)
IF(IC.GT.0)GOTO 290
WRITE(9,108)IDB,PRE,PLACE(IWHERE),E1,E2,LIST
108 FORMAT(A5,A6,A8/A8,A7/1X,A7)
1 READ (8,8,END=60)LINE
IF(COMP.NE.FROM)GOTO 1
8 FORMAT(10A8)
```

```
10 READ(8,8,END=60)LINE
IF(COMP.EQ.TO)GOTO 100
IF(CO.EQ.SS)GOTO 50
IF(CO.EQ.EE)GOTO 50
WRITE(9,6)LINE
6 FORMAT(1X,9A8)
GOTO 10
50 WRITE(9,55)
55 FORMAT(/)
GOTO 10
60 CALL CLOSE(8)
WRITE(NOUT,65) NAME
65 FORMAT(' JOB # OR FILE ',A8,' IS NOT FORMATTED CORRECTLY FOR OUTPUT')
CALL CLOSE(9)
GOTO 200
100 CALL CLOSE(8)
CALL CLOSE(9)
CALL SEND(ONAME,'U',IC)
IF(ISYS.EQ.1)CALL XPRESS(JOB,IC)
IF(ISYS.EQ.2)CALL HASP(JOB,IC)
CALL DEST(ONAME,PP)
IF(IC)110,140,170
110 WRITE(NOUT,115)
115 FORMAT(' RUN IS DISABLED ON PLORTS. PLEASE TRY AGAIN LATER.')
GOTO 200
140 WRITE(NOUT,145)JOB
145 FORMAT(' YOUR JOB # IS ',A8)
GOTO 200
170 CALL ERR(10001)
200 CALL DEST(ONAME,IC)
240 RETURN
290 CALL CLOSE(8)
CALL ERR(10003)
300 WRITE(NOUT,305)
305 FORMAT(' JOB # IS INCORRECT')
GOTO 240
END
"
"
"*****************************************************************
"* LPRINT DISPLAYS THE CONTENT OF A PRINT FILE BETWEEN
"* THE TWO FLAGS PASSED AS PARAMETERS. THE DIFFERENCE FROM
"* 'CALL PRINT' IS THE ELIMINATION DURING LPRINT OF
"* THE SECTION PRINT FLAGS LEFT IN THE OUTPUT BY SOUPEX.
"* THIS ROUTINE IS ONLY USED BY THE MAIN PROGRAM FOR LISTING
"* THE ENTIRE OUTPUT TO THE USER.
"*****************************************************************
SUBROUTINE LPRINT (NAME,FROM,TO,IC)
REAL*8 NAME,FROM,TO,COMP,LINE(9)
REAL*4 SS/'STA '/,EE/'END '/,CO
EQUIVALENCE (LINE(1),COMP,CO)
CALL OPEN (8,NAME,'INPUT',IC)
IF(IC)90,1,90
```

```
1 READ (8,8,END=60)LINE
IF(COMP.NE.FROM)GOTO 1
8 FORMAT(9A8)
10 READ(8,8,END=60)LINE
IF(COMP.EQ.TO)GOTO 40
IF(CO.EQ.SS)GOTO 100
IF(CO.EQ.EE)GOTO 100
WRITE(6,6)LINE
6 FORMAT(1X,9A8)
GOTO 10
40 CALL CLOSE(8)
RETURN
60 IC=-3
GOTO 40
90 IC=1
RETURN
100 WRITE(6,106)
106 FORMAT(1X)
GOTO 10
END
"
"
"*****************************************************************
"* ALFTOI IS USED TO TRANSULATE ALPHABETIC INPUT TO A NUMERIC
"* VALUE. AT ALL TIMES ALPHABETIC INTERUPT WORDS ARE VALID
"* RESPONSES AND READING THEM WITH AN INTEGER FORMAT WOULD
"* BE DISASTOROUS, HENCE THE NEED FOR THIS ROUTINE. THE
"* CHARACTER STRING IS PROVIDED IN THE VARIABLE IALF AND
"* THE NUMBER IS RETURNED IN IIN. IF THE RESPONSE IS NOT
"* A VALID NUMBER THEN AN ERROR RETURN IS TAKEN.
"*****************************************************************
SUBROUTINE ALFTOI(/IALF/,/IIN/,*)
INTEGER*2 IALF(8),ICALC(8),BLANK/' '/
INTEGER*2 DIGIT(10)/'0','1','2','3','4','5','6','7','8','9'/
DO 30 I=1,8
DO 20 J=1,10
IF(IALF(I).EQ.DIGIT(J))GOTO 25
20 CONTINUE
IF(IALF(I).NE.BLANK)GOTO 40
GOTO 50
25 L=J-1
ICALC(I)=L
30 CONTINUE
40 RETURN 1
50 L=I-1
IF(L.LT.1)GOTO 40
ISUM=0
DO 80 I=1,L
80 ISUM=ISUM*10+ICALC(I)
IIN=ISUM
RETURN
END
"
```

```
"
"*****************************************************************
"* STANR IS CALLED EVERY TIME A RESPONSE IS READ. IT COMPARES
"* ALL RESPONSES AGAINST THE SET OF INTERUPT WORDS. IF A MATCH
"* OCCURS, THE INTERUPT VALUE IS SET IN A VARIABLE IN COMMON
"* AND A RETURN 1 IS ISSUED. THE MAIN PROGRAM THEN PREFORMS
"* THE REQUIRED FUNCTION.
"*****************************************************************
SUBROUTINE STANR(*)
REAL*8 REPLY,REPLYR(24)
INTEGER*2 HALP,WHY,IALF(8)
EQUIVALENCE (REPLY,IALF(1))
COMMON IALF,HALP
DATA REPLYR/%
'R E S T','B R A N','B A C K','E X P L',%
'H E L P','E X P E','F A M I','N E W  ',%
'S T O P','S T E P','D J O B','F I L E',%
'R E V I','C J O B','M E S S','D A M N',%
'C O C K','A S S  ','C R A P','J O E  ',%
'C U N T','S H I T','F U C K','H E L L'/
DO 78 WHY=1,24
IF(REPLY.NE.REPLYR(WHY))GOTO 78
HALP=WHY
RETURN 1
78 CONTINUE
RETURN
END
"
"
"*****************************************************************
"* CCMES IS CALLED AFTER MOST OF CALL-OS'S INTERACTIVE FUNCTION
"* SUBROUTINES. THE TYPE OF SUBROUTINE CALL AND THE COMPLETION
"* CODE ARE PASSED AS PARAMETERS. IF AN ABNORMAL RETURN HAPPENS
"* THIS SUBROUTINE PRINTS A MESSAGE DESCRIBING THE CONDITION.
"*****************************************************************
SUBROUTINE CCMES(ITYPE,ICC)
REAL*8 TYPE(6)/'SEND','RUN','DEST','PRINT','OPEN','WAIT'/
COMPLEX *16 MSGS(6,6)/-
'TOO MANY FILES  ','FILE IN USE     ','                ','NO SUCH FILE
'CATALOG FILE    ','                ','RUN DISABLED    ','
'NO FILES TO RUN ','                ','                ','
'FILE IN USE     ','                ','NO SUCH FILE    ','
'                ','                ','EOF IN PRINT    ','NO LINES
'FILE IN USE     ','                ','NO SUCH FILE    ','CATALOG FILE
'NO SPACE        ','NO SUCH FILE    ','CATALOG FILE    ','FILE IN USE
'BAD FILENAME    ','                ','FILE IS FULL    ','OUT OF SPACE
'                ','OUT OF TIME     ','                ','
/
INTEGER IFACT(6)/3,2,2,4,0,3/
IF(ICC.EQ.0)RETURN
ISUM=ICC+IFACT(ITYPE)
WRITE(6,101)TYPE(ITYPE),MSGS(ISUM,ITYPE)
101 FORMAT(1X,A5,' EXIT INDICATES -- ',A16)
```

```
RETURN
END
"
"
"*************************************************************
"* THE EXIST SUBROUTINE TRIES TO OPEN THE FILENAME WHICH IS
"* PASSED TO IT.  IT IS USED TO VERIFY THE EXISTANCE OF FILES
"* WHOSE NAMES ARE TYPED IN BY THE USER.  IT CAN ALSO BE USED
"* TO MAKE SURE A FILE DOESN'T ALREADY EXIST WHEN THE USER
"* SUPPLIES THE NAME OF A FILE TO BE CREATED.
"*************************************************************
SUBROUTINE EXIST(FILE,*,*)
REAL*8 FILE
CALL OPEN(4,FILE,'INPUT',IC)
CALL CCMES(5,IC)
IF (IC)10,20,30
10 RETURN 1
20 CALL CLOSE(4)
RETURN
30 RETURN 2
END
"
"
"*************************************************************
"* THE BRANCH SUBROUTINE BRANCHES TO OTHER STATISTICAL SECTIONS
"* IN I.E.S.S.  THIS SUBROUTINE IS CURRENTLY NOT USED.  IT IS
"* DESIGNED TO BE COMPATIBLE WITH THE CHAINING FUNCTION IN
"* CALL-OS FORTRAN WHEN IT IS LOCALLY IMPLEMENTED.
"*************************************************************
SUBROUTINE BRANCH(IB)
REAL*8 SECT(4)/'IEFACTOR','IEREGRES','IEFREQUE','IELINEAR'/
CALL CHAIN(SECT(IB))
RETURN
END
"
"
"*************************************************************
"* THE CHAIN SUBROUTINE IS USED TO SIMMULATE THE CHAINING
"* FEATURE IN CALL-OS FORTRAN UNTIL IT IS IMPLEMENTED LOCALLY.
"*************************************************************
SUBROUTINE CHAIN(PROG)
REAL*8 PROG
WRITE (6,1) PROG
1 FORMAT (' TYPE ''EXEC ',A8,'''')
STOP 22
RETURN
END
"
"
"*************************************************************
"* THE ERR SUBROUTINE IS CALLED ONLY UNDER EXTREME CONDITIONS.
"* IT TERMINATES THE PROGRAM AND PRINTS AN ERROR CODE.
"* IT WAS MOSTLY USED IN THE INITIAL DEBUGGING OF THE PROGRAM.
```

```
"*****************************************************************
SUBROUTINE ERR(IERR)
INTEGER NOUT/6/
WRITE(NOUT,6)IERR
6 FORMAT(' ERROR # ',I5,' SEE SOUPAC CONSULTANT FOR EXPLANATION.')
STOP 8
RETURN
END
```

# Appendix A -- Common Message File

```
KIMKIM01
AT PRESENT, THIS IS THE ONLY SECTION OF I.E.S.S. AVAILABLE.
MORE SECTIONS WILL BE AVAILABLE SOON.
IF YOU WOULD LIKE TO CONTINUE WITH THIS SECTION, TYPE 'RESTART'.
IF NOT, TYPE 'STOP'.
KIMKIM02
AT PRESENT, THIS IS THE ONLY SECTION OF I.E.S.S. AVAILABLE.
MORE SECTIONS WILL BE AVAILABLE SOON.
IF YOU WOULD LIKE TO CONTINUE WITH THIS SECTION, TYPE 'RESTART'.
IF NOT, TYPE 'STOP'.
KIMKIM03
AT PRESENT, THIS IS THE ONLY SECTION OF I.E.S.S. AVAILABLE.
MORE SECTIONS WILL BE AVAILABLE SOON.
IF YOU WOULD LIKE TO CONTINUE WITH THIS SECTION, TYPE 'RESTART'.
IF NOT, TYPE 'STOP'.
KIMKIM04
IN ORDER TO RUN A PROGRAM, YOU MUST SUPPLY INPUT DATA.
THE DATA CAN EITHER BE CARD IMAGES STORED IN A PLORTS FILE,
OR CAN BE CREATED HERE AT THE TERMINAL.
IF YOUR DATA IS IN CARD IMAGE FORM, A FORMAT IS REQUIRED TO INDICATE
HOW MANY VARIABLES YOU HAVE AND WHERE THEY APPEAR ON THE CARDS.
THIS FORMAT CAN BE IN FRONT OF YOUR DATA IN THE SAME FILE, CAN BE ON
CARDS IN A DIFFERENT FILE, OR YOU CAN TYPE IT HERE AT THE TERMINAL.
IF YOUR FORMAT AND DATA ARE TOGETHER IN THE SAME FILE, TYPE 'S'.
IF YOUR FORMAT AND DATA EXIST IN DIFFERENT FILES, TYPE 'D'.
IF YOUR DATA IS FILED AND YOU WANT TO TYPE YOUR FORMAT NOW, TYPE 'T'.
TO CREATE YOUR DATA NOW AT THE TERMINAL, TYPE 'C'.
KIMKIM05
IF YOUR FORMAT AND DATA ARE TOGETHER IN THE SAME FILE, TYPE 'S'.
IF YOUR FORMAT AND DATA EXIST IN DIFFERENT FILES, TYPE 'D'.
IF YOUR DATA IS FILED AND YOU WANT TO TYPE YOUR FORMAT NOW, TYPE 'T'.
TO CREATE YOUR DATA NOW AT THE TERMINAL, TYPE 'C'.
KIMKIM06
FORMAT AND DATA.
SAME FILE.
DIFFERENT FILES.
TYPE IN FORMAT.
CREATE BOTH AT TERMINAL.  S,D,T,C
KIMKIM07
ALSO, A FILE MUST EXIST CONTAINING YOUR FORMAT IN THE FORM:
DATA(# OF VARIABLES)(FORTRAN FORMAT).
TYPE THE NAME OF THAT FILE.
KIMKIM08
TYPE THE NAME OF THE FILE WHICH CONTAINS YOUR FORMAT.
KIMKIM09
FORMAT FILE NAME.
KIMKIM10
A FILE MUST ALREADY EXIST WHICH CONTAINS YOUR DATA IN CARD IMAGE FORM.
TYPE THE NAME OF THAT FILE.
KIMKIM11
TYPE THE NAME OF THE FILE WHICH CONTAINS YOUR DATA.
```

```
KIMKIM12
DATA FILE NAME.
KIMKIM13
A FILE MUST ALREADY EXIST WHICH CONTAINS YOUR FORMAT ON THE FIRST CARD,
FOLLOWED BY YOUR DATA CARDS.
THE FORMAT ON THE FIRST CARD SHOULD APPEAR IN THE FORM:
DATA (# OF VARIABLES) (FORTRAN FORMAT).
TYPE THE NAME OF THAT FILE.
KIMKIM14
TYPE IN THE NAME OF THE FILE WHICH CONTAINS YOUR FORMAT AND DATA.
KIMKIM15
FORMAT-DATA FILE NAME.
KIMKIM16
AT THIS TIME YOU MUST TYPE YOUR FORMAT ON ONE LINE USING THE FORM
(# OF VARIABLES) ( A FORTRAN FORMAT ).
EXAMPLES:
FOR 20 VARIABLES IN THE FIRST 20 COLUMNS TYPE  (20)(20F1.0).
FOR 3 VAR. IN COLS(1-2),COLS(3-6),AND COL(7) TYPE  (3)(F2.0,F4.0,F1.0).
FOR A DETAILED LESSON IN FORMAT TYPE 'FORMAT', OR
TYPE IN YOUR FORMAT AS INDICATED ABOVE.
 ( # ) ( FORTRAN FORMAT )
KIMKIM17
TYPE IN YOUR FORMAT ON ONE LINE USING THE FOLLOWING FORM.
(# OF VARIABLES) ( A FORTRAN FORMAT )
KIMKIM18
(#)(FORTRAN FORMAT).
KIMKIM19
BEFORE YOU TYPE IN YOUR FORMAT WHICH INDICATES HOW YOUR DATA APPEARS
ON YOUR DATA CARDS, WOULD YOU LIKE TO HAVE A FEW ROWS OF YOUR DATA
LISTED HERE AT THE TERMINAL. TYPE 'YES' OR 'NO'.
KIMKIM20
WOULD YOU LIKE TO SEE A FEW ROWS OF YOUR DATA. TYPE 'YES' OR 'NO'.
KIMKIM21
SEE FEW ROWS OF DATA. Y,N
KIMKIM22
YOUR JOB IS STILL AWAITING EXECUTION.
IF YOU WANT TO CONTINUE WAITING TYPE 'C'.
IF YOU DO NOT WANT TO WAIT, YOU CAN RETURN LATER AND REVIEW YOUR OUTPUT.
TO RETURN LATER TYPE 'R'.
KIMKIM23
YOUR JOB IS STILL AWAITING EXECUTION.
IF YOU WANT TO CONTINUE WAITING TYPE 'C'.
IF YOU WANT TO RETURN LATER TO REVIEW YOUR OUTPUT TYPE 'R'.
KIMKIM24
CONTINUE WAITING OR RETURN LATER. C,R
KIMKIM25
YOUR SOUPEX RUN HAS COMPLETED AND THE OUTPUT HAS BEEN FILED IN YOUR
PLORTS FILES.  IF YOU ARE RUNNING FROM A TELETYPE, TYPE 'E' AT THIS
TIME TO HAVE THE OUTPUT LISTED.  IF YOU ARE AT A DISPLAY TERMINAL,
TYPE 'E' TO REVIEW THE ENTIRE OUTPUT OR TYPE 'S' TO REVIEW THE
OUTPUT BY SECTIONS.
KIMKIM26
THE OUTPUT HAS BEEN FILED IN YOUR PLORTS FILES.
```

```
TO SEE THE ENTIRE OUTPUT, TYPE 'E'.
TO SEE PARTICULAR SECTIONS, TYPE 'S'.
KIMKIM27
SEE ENTIRE OUTPUT OR SECTIONS. E,S
KIMKIM28
DO YOU WANT A PRINTED COPY OF YOUR OUTPUT.
IT CAN BE ROUTED TO A TERMINAL CLOSE TO YOU AND RUN ON HASP OR EXPRESS.
TYPE 'YES' OR 'NO'.
KIMKIM29
DO YOU WANT A PRINTED COPY OF YOUR OUTPUT. TYPE 'YES' OR 'NO'.
KIMKIM30
PRINTED COPY. Y,N
KIMKIM31
IF YOU ARE NEAR A TERMINAL AND CAN PICK UP THE PRINTED COPY JOB
IMMEDIATELY, THEN YOU SHOULD RUN ON EXPRESS BY TYPING 'E'.
IF YOU HAVE OVER 300 LINES OR WANT TO PICK THE JOB UP LATER,
THEN YOU SHOULD RUN ON HASP BY TYPING 'H'.
KIMKIM32
TO RUN COPY JOB ON HASP, TYPE 'H'.
TO RUN IT ON EXPRESS, TYPE 'E'.
KIMKIM33
EXPRESS OR HASP. E,H
KIMKIM34
THE COPY OF YOUR OUTPUT CAN BE PRINTED AT THE FOLLOWING AREAS.
TO PRINT AT DCL. (CSO NORTH), TYPE 'LOCAL'.
TO PRINT AT COMM. WEST. (CSO SOUTH), TYPE 'COM'.
TO REMOTE TERMINALS, TYPE 'PSYCH', 'AGRIC', 'CIVIL', OR 'CHEM'.
KIMKIM35
TO PRINT AT CSO NORTH, TYPE 'LOCAL'.
TO PRINT AT CSO SOUTH, TYPE 'COMM'.
TO REMOTE TERMINALS, TYPE 'PSYCH', 'AGRIC', 'CIVIL', OR 'CHEM'.
KIMKIM36
WHICH TERMINAL. LOCAL,COMM,PSYCH,AGRIC,CIVIL,CHEM
KIMKIM37
THE OUTPUT FROM YOUR JOB EXISTS AS A FILE UNDER YOUR PS #.
IF YOU WANT TO KEEP THAT FILE FOR FUTURE USE, TYPE 'K'.
IF YOU WANT TO DESTROY THE FILE TO CONSERVE SPACE, TYPE 'D'.
KIMKIM38
IF YOU WANT TO KEEP THE FILED OUTPUT, TYPE 'K'.
IF YOU WANT TO DESTROY THE FILED OUTPUT, TYPE 'D'.
KIMKIM39
KEEP OR DESTROY FILED OUTPUT. K,D
KIMKIM40
IN ORDER TO REVIEW AN EXISTING OUTPUT, YOU MUST KNOW THE 8 CHARACTER
JOB NUMBER ASSIGNED TO IT.  TYPE THAT # WHEN THE QUESTION MARK APPEARS.
IF THE FILE DOES NOT EXIST, THE JOB HAS NOT FINISHED EXECUTION OR
IT HAS BEEN DESTROYED IN YOUR FILES.
KIMKIM41
TYPE IN THE 8 CHARACTER JOB # FOR THE OUTPUT YOU WOULD LIKE TO REVIEW.
KIMKIM42
JOB #.
KIMKIM43
AT THIS STAGE YOU HAVE THE OPTION OF RUNNING THE EXACT SAME ANALYSIS
```

```
WHILE CHANGING THE INPUT DATA AND/OR VARIABLE FORMAT.
WOULD YOU LIKE TO DO THAT. TYPE 'YES' OR 'NO'.
KIMKIM44
DO YOU WANT TO RUN THE SAME ANALYSIS, CHANGING ONLY THE DATA OR FORMAT.
TYPE 'YES' OR 'NO'.
KIMKIM45
SAME ANALYSIS, CHANGED DATA. Y,N
KIMKIM46
DO YOU WANT TO STOP NOW OR CONTINUE. TYPE 'STOP' OR 'CONT'.
KIMKIM47
STOP OR CONTINUE. TYPE 'S' OR 'C'.
KIMKIM48
STOP OR CONTINUE. S,C
KIMKIM49
YOU HAVE NOW COMPLETED THE CYCLE THRU THIS STATISTICAL SECTION.
TO SET OPTIONS FOR A DIFFERENT RUN, TYPE 'S'.
TO PICK A DEFAULT PATH WITH PRECHOSEN OPTIONS, TYPE 'D'.
TO REVIEW AN EXISTING FILED OUTPUT, TYPE 'R'.
TO BRANCH TO A DIFFERENT STATISTICAL SECTION, TYPE 'B'.
KIMKIM50
TO SET OPTIONS FOR ANOTHER RUN, TYPE 'S'.
TO CHOOSE A DEFAULT PATH WITH PRECHOSEN OPTIONS, TYPE 'D'.
TO REVIEW AN EXISTING FILED OUTPUT, TYPE 'R'.
TO BRANCH TO A DIFFERENT STATISTICAL SECTION, TYPE 'B'.
KIMKIM51
SET, DEFAULT, REVIEW, OR BRANCH TO ANOTHER STATISTICAL SECTION. S,D,R,B
KIMKIM52
YOU WILL ENTER YOUR DATA 1 OBSERVATION AT A TIME.
EACH OBSERVATION MUST HAVE THE SAME # OF VARIABLES.
HOW MANY VARIABLES DO YOU WISH TO HAVE.
KIMKIM53
HOW MANY VARIABLES WILL YOU HAVE PER OBSERVATION.
KIMKIM54
# OF VARIABLES.
KIMKIM55
WHAT IS THE LARGEST # OF DIGITS ANY OF YOUR VARIABLES WILL CONTAIN.
THIS INCLUDES LEADING MINUSES AND DECIMAL POINTS.
EXAMPLE. -567.123 CONTAINS 8 DIGITS.
KIMKIM56
WHAT IS THE MAXIMUM # OF DIGITS YOU WILL HAVE PER VARIABLE.
KIMKIM57
MAX. # OF DIGITS/VAR.
KIMKIM58
THE DATA WHICH YOU CREATE WILL BE PLACED INTO A FILE UNDER YOUR PS #.
THIS FILE WILL CONSIST OF A FORMAT CARD FOLLOWED BY YOUR DATA.
THE FILE WILL STILL EXIST AFTER THIS RUN.
YOU MAY REFER TO IT LATER AS A FORMAT-DATA FILE, OR DELETE IT.
SPECIFY A NAME FOR THE FILE TO CONTAIN YOUR CREATED DATA.
KIMKIM59
SPECIFY A NAME FOR THE FILE TO CONTAIN YOUR CREATED DATA.
IT WILL BE CREATED AND KEPT AS A FILE UNDER YOUR PS #.
KIMKIM60
NAME OF FILE TO PLACE DATA.
```

```
KIMKIM61
EACH TIME A QUESTION MARK APPEARS ENTER 1 LINE OR OBSERVATION OF DATA.
SEPARATE YOUR VARIABLES WITHIN EACH OBSERVATION BY BLANKS.
EXAMPLE:  12  34.5 -78  .99999 -12 34.5  1  98765
AFTER THE LAST OBSERVATION OF DATA, TYPE 'EOF'.
KIMKIM62
CREATE YOUR DATA ONE LINE AT A TIME.
SEPARATE VARIABLES BY BLANKS AND TYPE 'EOF' AFTER THE LAST OBSERVATION.
KIMKIM63
CREATE DATA.
KIMKIM64
THE PARAMETERS AND DATA HAVE BEEN SET UP FOR A SOUPEX JOB RUN.
TO RUN YOUR JOB ON EXPRESS, TYPE 'E'.
TO RUN YOUR JOB ON HASP, TYPE 'H'.
TO BYPASS RUNNING AT THIS TIME, TYPE 'B'.
KIMKIM65
TO RUN YOUR JOB ON EXPRESS, TYPE 'E'.
TO RUN YOUR JOB ON HASP, TYPE 'H'.
TO BYPASS RUNNING AT THIS TIME, TYPE 'B'.
KIMKIM66
RUN JOB ON EXPRESS, HASP, OR BYPASS. E,H,B
KIMKIM67
WOULD YOU LIKE TO SEE MORE OF YOUR DATA. TYPE 'YES' OR 'NO'.
KIMKIM68
SEE MORE OF YOUR DATA. TYPE 'YES' OR 'NO'.
KIMKIM69
MORE. Y,N
KIMKIM70
END OF FILE
```

## Appendix B -- Correlations-Factor Analysis Message File

```
KIMKIM01
THE CORRELATIONS PROGRAM CALCULATES MEANS AND STANDARD DEVIATIONS
OF YOUR VARIABLES. IT THEN CALCULATES THE FOLLOWING MATRICES.
CROSS-PRODUCTS MATRIX, COVARIANCE MATRIX, AND CORRELATION MATRIX.
DO YOU WANT TO SEE THE MEANS AND STANDARD DEVIATIONS.
TYPE 'YES' OR 'NO'.
KIMKIM02
DO YOU WANT MEANS AND STANDARD DEVIATIONS. TYPE 'YES' OR 'NO'.
KIMKIM03
MEANS AND ST.DEV. Y,N
KIMKIM04
THE CORRELATIONS MATRIX CONSISTS OF PEARSON PRODUCT-MOMENT
CORRELATION COEFFICIENTS BETWEEN ALL PAIRS OF VARIABLES,
ARRANGED IN A SQUARE SYMMETRIC MATRIX.
DO YOU WANT TO SEE THE CORRELATION MATRIX.
TYPE 'YES' OR 'NO'.
KIMKIM05
DO YOU WANT TO SEE THE CORRELATION MATRIX. TYPE 'YES' OR 'NO'.
KIMKIM06
CORRELATIONS. Y,N
KIMKIM07
THE COVARIANCE MATRIX IS A SQUARE SYMETRIC MATRIX WITH DIAGONAL ELEMENTS
SPECIFING THE VARIANCE OF EACH VARIABLE AND OFF-DIAGONAL ELEMENTS
SPECIFING THE COVARIANCE BETWEEN ALL POSSIBLE PAIRS OF VARIABLES.
THE VARIANCE OF A VARIABLE IS THE SUM OF THE SQUARED DEVIATIONS FROM
THE MEAN, DIVIDED BY N.  THE COVARIANCE BETWEEN A PAIR OF VARIABLES
IS THE SUM OF PRODUCTS OF DEVIATION SCORES ON THE TWO VARIABLES,
DIVIDED BY N.
DO YOU WANT TO SEE THE COVARIANCE MATRIX.
TYPE 'YES' OR 'NO'.
KIMKIM08
DO YOU WANT TO SEE THE COVARIANCE MATRIX. TYPE 'YES' OR 'NO'.
KIMKIM09
COVARIANCES. Y,N
KIMKIM10
THE CROSS-PRODUCTS MATRIX IS A SQUARE SYMETRIC MATRIX WITH DIAGONAL
ELEMENTS SPECIFYING THE SUM OF SQUARES OF THE OBSERVED SCORES ON EACH
VARIABLE AND OFF-DIAGONAL ELEMENTS SPECIFYING THE SUM OF PRODUCTS OF
THE OBSERVED SCORES FOR ALL POSSIBLE PAIRS OF VARIABLES.
DO YOU WANT TO SEE THE CROSS-PRODUCTS MATRIX.
TYPE 'YES' OR 'NO'.
KIMKIM11
DO YOU WANT TO SEE THE CROSS-PRODUCTS MATRIX. TYPE 'YES' OR 'NO'.
KIMKIM12
CROSS PRODUCTS. Y,N
KIMKIM13
KIMKIM22
DO YOU WANT TO FACTOR ANALYZE ONE OF THE MATRICES CALCULATED IN
THE CORRELATION PROGRAM. TYPE 'YES' OR 'NO'.
KIMKIM23
DO YOU WANT TO DO FACTOR ANALYSIS. TYPE 'YES' OR 'NO'.
```

```
KIMKIM24
FACTOR ANALYSIS. Y,N
KIMKIM25
WHICH MATRIX DO YOU WANT TO FACTOR.
FOR CORRELATION MATRIX, TYPE 'COR'.
FOR COVARIANCE MATRIX, TYPE 'COV'.
FOR CROSS-PRODUCTS MATRIX, TYPE 'CRO'.
KIMKIM26
WHICH MATRIX DO YOU WANT TO FACTOR.
TYPE 'COR', 'COV', OR 'CRO'.
KIMKIM27
FACTOR WHICH MATRIX. COR,COV,CRO
KIMKIM28
THE VARIANCE OF EACH OBSERVED VARIABLE IS MADE UP OF TWO PARTS.
ONE PART IS THAT WHICH IS IN COMMON WITH THE OTHER VARIABLES IN THE SET,
OR WHICH CAN BE ACCOUNTED FOR BY VARIATION IN THE OTHER VARIABLES IN THE
SET. A SECOND PART IS THAT WHICH IS UNIQUE TO EACH VARIABLE, OR WHICH
CANNOT BE ACCOUNTED FOR BY VARIATION IN THE OTHER VARIABLES IN THE SET.
ESTIMATES OF THE COMMON VARIANCE OF EACH VARIABLE ARE CALLED COMMUNALITY
ESTIMATES.  THE GENERAL FACTOR ANALYSIS MODEL, WHICH INVOLVES AN
EXAMINATION OF THE UNDERLYING RELATIONSHIPS AMONG THE VARIABLES, REQUIRE
THAT COMMUNALITY ESTIMATES BE ENTERED IN THE DIAGONAL OF THE CORRELATION
MATRIX.  IF THIS IS NOT DONE, THE USER IS DOING PRINCIPAL COMPONENTS
ANALYSIS, WHICH IS A SPECIAL CASE OF THE GENERAL FACTOR ANALYSIS MODEL.
DO YOU WANT COMMUNALITY ESTIMATION. TYPE 'YES' OR 'NO'.
KIMKIM29
DO YOU WANT COMMUNALITY ESTIMATION. TYPE 'YES' OR 'NO'.
KIMKIM30
COMMUNALITIES. Y,N
KIMKIM31
CHOOSE METHOD OF ESTIMATION.
FOR LARGEST ABSOLUTE ROW CORRELATION, TYPE 'L'.
FOR MULTIPLE R SQUARED, TYPE 'M'.
FOR SQUARE ROOT OF AVERAGE SQUARE, TYPE 'S'.
KIMKIM32
CHOOSE METHOD OF ESTIMATION.
FOR LARGEST ABSOLUTE ROW CORRELATION, TYPE 'L'.
FOR MULTIPLE R SQUARED, TYPE 'M'.
FOR SQUARE ROOT OF AVERAGE SQUARE, TYPE 'S'.
KIMKIM33
ESTIMATION. L,M,S
KIMKIM34
TYPE THE MAXIMUM # OF FACTORS TO BE EXTRACTED.
THIS NUMBER MUST BE LESS THAN OR EQUAL TO # OF VARIABLES.
TO ELIMINATE THIS CONSTRAINT, TYPE '0'.
KIMKIM35
TYPE THE MAXIMUM # OF FACTORS TO BE EXTRACTED.
KIMKIM36
MAX. # OF FACTORS.
KIMKIM37
TYPE THE % OF TOTAL VARIANCE TO BE REMOVED.
THE # MUST BE LESS THAN OR EQUAL TO 100.
TO ELIMINATE THIS CONSTRAINT, TYPE '0'.
```

```
KIMKIM38
TYPE THE % OF TOTAL VARIANCE TO BE REMOVED.
KIMKIM39
% OF VARIANCE.
KIMKIM40
DO YOU WANT FACTORING TO STOP WHEN CONTRIBUTIONS FALL BELOW UNITY.
TYPE 'YES' OR 'NO'.
KIMKIM41
DO YOU WANT FACTORING TO STOP WHEN CONTRIBUTIONS FALL BELOW UNITY.
TYPE 'YES' OR 'NO'.
KIMKIM42
STOP IF EIGS. ¢ UNITY. Y,N
KIMKIM43
DO YOU WANT TO ROTATE YOUR FACTOR MATRIX USING VARIMAX ROTATION.
TYPE 'YES' OR 'NO'.
KIMKIM44
DO YOU WANT A VARIMAX ROTATION. TYPE 'YES' OR 'NO'.
KIMKIM45
VARIMAX. Y,N
KIMKIM46
IN RAW VARIMAX, THE UNCHANGED INPUT FACTOR MATRIX IS ROTATED BY THE
VARIMAX CRITERION.  HOWEVER, BECAUSE THE VARIABLES USUALLY HAVE
DIFFERENT COMMUNALITIES, THE LOADINGS ON A FACTOR WILL HAVE SPURIOUS
DIFFERENTIAL WEIGHTINGS.  THEREFORE, IN THE NORMAL VARIMAX PROCEDURE
THE RELATIVE WEIGHTS OF THE VARIABLES ARE EQUALIZED BY SCALING ALL
VARIABLES TO HAVE UNIT VARIANCES; ROTATED ACCORDING TO THE VARIMAX
CRITERION; AND THEN RESCALED BACK TO THE ORIGINAL VARIANCES.
     IT HAS BEEN SHOWN THAT THIS PROCEDURE TENDS TOWARD BETTER
FACTORIAL INVARIANCE THAN THE RAW VARIMAX. FACTORIAL INVARIANCE IS
WHEN THE FACTORIAL DESCRIPTION OF A VARIABLE REMAINS INVARIANT WHEN
THE VARIABLE IS FACTOR ANALYZED WITH A DIFFERENT SET OF VARIABLES
INVOLVING THE SAME COMMON FACTOR.
FOR A NORMAL VARIMAX, TYPE 'N'.
FOR A RAW VARIMAX, TYPE 'R'.
KIMKIM47
FOR A NORMAL VARIMAX, TYPE 'N'.
FOR A RAW VARIMAX, TYPE 'R'.
KIMKIM48
NORMAL OR RAW. N,R
KIMKIM49
DO YOU WANT THE TRANSFORMATION MATRIX T CALCULATED.
TYPE 'YES' OR 'NO'.
KIMKIM50
DO YOU WANT THE TRANFORMATION MATRIX. TYPE 'YES' OR 'NO'.
KIMKIM51
T-MATRIX. Y,N
KIMKIM52

SECTIONS MAY BE VIEWED BY TYPING:
'MEANS', 'CORR', 'COVA', 'CROS', 'COMM', 'PRIN', 'VARI', OR 'ENTIRE'.
DO NOT ASK FOR A SECTION THAT WAS NOT CALCULATED.
WHEN NO SECTIONS ARE DESIRED, TYPE 'DONE'.
KIMKIM53
```

```
SECTIONS MAY BE VIEWED BY TYPING:
'MEANS', 'CORR', 'COVA', 'CROS', 'COMM', 'PRIN', 'VARI', OR 'ENTIRE'.
TYPE 'DONE' TO EXIT.
KIMKIM54

SECTION.
KIMKIM55
TYPE ONE OF THE OPTION NUMBERS. THIS WILL CAUSE THE EXECUTION OF A
PREDESCRIBED SOUPEX RUN AFTER YOU SELECT YOUR DATA.
OPTION #                    DESCRIPTION
  1          COR. ONLY WITH ALL OPTIONS.
  2          COR,PRI,VAR WITH MAX # FACTORS AND 100% VARIANCE.
KIMKIM56
OPTION #                    DESCRIPTION
  1          COR. ONLY WITH ALL OPTIONS.
  2          COR,PRI,VAR WITH MAX # FACTORS AND 100% VARIANCE.
KIMKIM57
TYPE '1' FOR COR. ONLY.
TYPE '2' FOR COR-PRI-VAR.
KIMKIM58
THIS IS THE CORRELATIONS-FACTOR ANALYSIS SECTION OF I.E.S.S.
TYPE AN 'S' AT THIS TIME IN ORDER TO SET PARAMETERS FOR A
CORRELATIONS-FACTOR ANALYSIS RUN ON THE EXPRESS SYSTEM.
IF YOU DO NOT WANT TO CONTINUE, TYPE THE WORD 'STOP' AND HIT RETURN.
KIMKIM59
WELCOME TO THE CORRELATIONS-FACTOR ANALYSIS SECTION OF I.E.S.S.
TO USE I.E.S.S. EFFICIENTLY, NEVER TYPE ANYTHING UNTIL A QUESTION MARK
APPEARS AND HAVE YOUR I.E.S.S. GUIDE SHEET AVAILABLE FOR REFERENCE.
FOR DOCUMENTATION AND QUESTIONS, SEE THE SOUPAC CONSULTANTS, 84 COM WEST

TO SET OPTIONS FOR A CORRELATIONS-FACTOR ANALYSIS RUN, TYPE 'S'.
TO CHOOSE A DEFAULT PATH WITH PRECHOSEN OPTIONS, TYPE 'D'.
TO REVIEW AN EXISTING FILED OUTPUT, TYPE 'R'.
KIMKIM60
SET, DEFAULT, OR REVIEW. S,D,R
KIMKIM61
END OF FILE
```

# Appendix C -- Preliminary I.E.S.S. Guide

## I. INTRODUCTION

I.E.S.S. (Interactive Express Statistic System) is a system of interactive Call-OS Fortran programs resident on the University of Illinois PLORTS timesharing system. I.E.S.S. interacts with the user in choosing options and data for a chosen statistical analysis. It then submits a SOUPEX job to be run on express, retrieves the results, and displays them to the user. This procedure can be repeated many times during a single sitting at the terminal while changing data, options, or statistical analysis on each SOUPEX run.

This system is designed for class use, where small samples of data are being analyzed, and as a research tool for producing preliminary descriptive statistics before full scale analysis is attempted.

## II. STATISTICS

The preliminary I.E.S.S. system contains only the correlation-factor analysis statistical section. Available in this section are the following statistical techniques:

correlation analysis
communality estimation
principal axis factor analysis
varimax rotation

Statistical sections to be added at a later date include regression analysis, linear programming, and one way frequency counts.

## III. DATA

Input data can be created at the terminal during your I.E.S.S. session for samples with a small number of variables. This created data is kept in a PLORTS file and can be reused during a later session. Much more flexibility can be gained by having your data stored in card image form in a PLORTS file before your I.E.S.S. session. This eliminates typing the data at the terminal; allows a greater number of variables by using multi-card observations; and also allows different subsets of variables to be analyzed during successive SOUPEX runs. In order for the user to specify in which columns of the card his variables appear, a prior knowledge of Fortran formats is recommended.

## IV. ACCESS

In order to use I.E.S.S., you must have a PLORTS sign-on and permission for PLORTS disk space. After logging in, all you must do is type )FACTOR followed by CR to execute the correlations-factor analysis section of I.E.S.S. Due to the fact that I.E.S.S. is a large program, it will take more than a few seconds to load. Be patient and avoid hitting the return key while waiting.

## V. RESPONSES

Each time a message is displayed at your terminal, a response will be required. Most messages will include a set of possible responses from which to choose. Your first response will indicate which of three general paths through I.E.S.S. you would like to follow. On your initial use of I.E.S.S., you should type an 'S', which indicates that you would like to set all parameters for the SOUPEX run. This will help to acquaint you with what options are available. After your initial use, you may find the other paths convenient. The default path option allows you to choose one of two preset sequences of parameter options. The job review path allows you to look at filed output kept from earlier SOUPEX runs through I.E.S.S.

Messages which don't list a set of responses will usually require a numerical or filename response. I.E.S.S. will verify the validity of numerics within reason bounds and will also verify the correct status of specified filenames.

The third type of response can be one of the following 'interrupt' words. They can be typed instead of a response of either of the above types. They will temporarily interrupt the program flow to achieve the specified function.

| *WORD TO TYPE* | *FUNCTION PERFORMED* |
|---|---|
| BACK | Backs up to the previous question. |
| HELP | In most cases will give a more detailed explanation for what is being asked. |
| RESTART | Will return to the beginning and start over. |
| STOP | Terminates your I.E.S.S. Session. |
| EXPERIENCED | Puts you in short abbreviated message mode. If you type 'HELP' or 'EXPLAIN', you automatically leave short mode. |
| FILE | Allows you to have listed a file of your choice from your PLORTS files. |
| MESSAGE | If a file named IESSMESS exists in your files, it will be listed at that time. |
| REVIEW | Will immediately branch to the section of I.E.S.S. which reviews previously filed output. |
| DJOB | Displays the current contents of the job name pointer, which after a SOUPEX run contains the name (OS job #) of the PLORTS file in which the output is stored. |
| CJOB | Allows you to change the contents of the job name pointer. This is used to compare sections of outputs from different SOUPEX runs. |

## Appendix D -- Sample Session at Terminal

(ALL COMMANDS PRECEDED BY * WERE TYPED BY THE USER)

```
*)FACTOR

 WELCOME TO THE CORRELATIONS-FACTOR ANALYSIS SECTION OF I.E.S.S.
 TO USE I.E.S.S. EFFICIENTLY, NEVER TYPE ANYTHING UNTIL A QUESTION MARK
 APPEARS AND HAVE YOUR I.E.S.S. GUIDE SHEET AVAILABLE FOR REFERENCE.
 FOR DOCUMENTATION AND QUESTIONS, SEE THE SOUPAC CONSULTANTS.

 TO SET OPTIONS FOR A CORRELATIONS-FACTOR ANALYSIS RUN, TYPE 'S'.
 TO CHOOSE A DEFAULT PATH WITH PRECHOSEN OPTIONS, TYPE 'D'.
 TO REVIEW AN EXISTING FILED OUTPUT, TYPE 'R'.
 ?
*SET

 DO YOU WANT MEANS AND STANDARD DEVIATIONS. TYPE 'YES' OR 'NO'.
 ?
*YES

 DO YOU WANT TO SEE THE CORRELATION MATRIX. TYPE 'YES' OR 'NO'.
 ?
*YES

 DO YOU WANT TO SEE THE COVARIANCE MATRIX. TYPE 'YES' OR 'NO'.
 ?
*SKIP

 IF YOUR FORMAT AND DATA ARE TOGETHER IN THE SAME FILE, TYPE 'S'.
 IF YOUR FORMAT AND DATA EXIST IN DIFFERENT FILES, TYPE 'D'.
 IF YOUR DATA IS FILED AND YOU WANT TO TYPE YOUR FORMAT NOW, TYPE 'T'.
 TO CREATE YOUR DATA NOW AT THE TERMINAL, TYPE 'C'.
 ?
*SAME

 TYPE IN THE NAME OF THE FILE WHICH CONTAINS YOUR FORMAT AND DATA.
 ?
*TESTDATA

 DO YOU WANT TO DO FACTOR ANALYSIS. TYPE 'YES' OR 'NO'.
 ?
*YES

 WHICH MATRIX DO YOU WANT TO FACTOR.
 TYPE 'COR', 'COV', OR 'CRO'.
 ?
*COR

 DO YOU WANT COMMUNALITY ESTIMATION. TYPE 'YES' OR 'NO'.
 ?
*NO
```

```
 TYPE THE MAXIMUM # OF FACTORS TO BE EXTRACTED.
 ?
*3

 TYPE THE % OF TOTAL VARIANCE TO BE REMOVED.
 ?
*100

 DO YOU WANT FACTORING TO STOP WHEN CONTRIBUTIONS FALL BELOW UNITY.
 TYPE 'YES' OR 'NO'.
 ?
*NO

 DO YOU WANT A VARIMAX ROTATION. TYPE 'YES' OR 'NO'.
 ?
*NO

 TO RUN YOUR JOB ON EXPRESS, TYPE 'E'.
 TO RUN YOUR JOB ON HASP, TYPE 'H'.
 TO BYPASS RUNNING AT THIS TIME, TYPE 'B'.
 ?
*EXPRESS

 YOUR JOB HAS BEEN SUBMITTED FOR EXECUTION.
 WE ARE NOW WAITING FOR THE OUTPUT.

 THE OUTPUT HAS BEEN FILED IN YOUR PLORTS FILES.
 TO SEE THE ENTIRE OUTPUT, TYPE 'E'.
 TO SEE PARTICULAR SECTIONS, TYPE 'S'.
 ?
*SECTIONS

 SECTIONS MAY BE VIEWED BY TYPING:
 'MEANS', 'CORR', 'COVA', 'CROS', 'COMM', 'PRIN', 'VARI', OR 'ENTIRE'.
 TYPE 'DONE' TO EXIT.
 ?
*MEANS

        MEAN              STANDARD DEVIATION

   1    0.2920000D 02     0.1210620D 02
   2    0.3780000D 02     0.1991381D 02
   3    0.7740000D 02     0.1488086D 02


 SECTIONS MAY BE VIEWED BY TYPING:
 'MEANS', 'CORR', 'COVA', 'CROS', 'COMM', 'PRIN', 'VARI', OR 'ENTIRE'.
 TYPE 'DONE' TO EXIT.
 ?
*COR
```

```
 CORRELATION MATRIX
           1          2          3
   1     1.0000
   2     0.1661     1.0000
   3     0.6668     0.7487     1.0000


 SECTIONS MAY BE VIEWED BY TYPING:
 'MEANS', 'CORR', 'COVA', 'CROS', 'COMM', 'PRIN', 'VARI', OR 'ENTIRE'.
 TYPE 'DONE' TO EXIT.
 ?
*PRI

 FACTOR      VARIANCE           PERCENT VARIANCE     CUMULATIVE PERCENT

      1       2.0886               69.6212               69.6212
      2       0.8351               27.8363               97.4575
      3       0.0763                2.5425              100.0000
 THE TRACE IS       3.0000
 THE SUM OF THE FIRST     3 ROOTS IS       3.0000

 COLUMN FACTOR MATRIX
           1          2          3
   1     0.7191     0.6843     0.1209
   2     0.7830    -0.6053     0.1429
   3     0.9790    -0.0185    -0.2031


 SECTIONS MAY BE VIEWED BY TYPING:
 'MEANS', 'CORR', 'COVA', 'CROS', 'COMM', 'PRIN', 'VARI', OR 'ENTIRE'.
 TYPE 'DONE' TO EXIT.
 ?
*DONE

 DO YOU WANT A PRINTED COPY OF YOUR OUTPUT. TYPE 'YES' OR 'NO'.
 ?
*NO

 IF YOU WANT TO KEEP THE FILED OUTPUT, TYPE 'K'.
 IF YOU WANT TO DESTROY THE FILED OUTPUT, TYPE 'D'.
 ?
*DESTROY

 DO YOU WANT TO RUN THE SAME ANALYSIS, CHANGING ONLY THE DATA OR FORMAT.
 TYPE 'YES' OR 'NO'.
 ?
*NO

 STOP OR CONTINUE. TYPE 'S' OR 'C'.
 ?
*STOP

 STOPPED AFTER EXECUTING 24 STEPS.
```

| BIBLIOGRAPHIC DATA SHEET | 1. Report No. UIUCDCS-R-74-653 | 2. | 3. Recipient's Accession No. |
|---|---|---|---|

4. Title and Subtitle
Interactive Express Statistical System

5. Report Date
June 1974

6.

7. Author(s)
William Charles Walter

8. Performing Organization Rept. No. UIUCDCS-R-74-653

9. Performing Organization Name and Address
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.

12. Sponsoring Organization Name and Address
Computing Services Office and Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

13. Type of Report & Period Covered
Master of Science Thesis

14.

15. Supplementary Notes

16. Abstracts

The need for easy to use statistical systems has grown rapidly in recent years. Interactive Express Statistical System (I.E.S.S.) combines the fast turnaround of the Express job system with the conversation capabilities of Call-OS Fortran on the PLORTS time-sharing system. It provides an instructive and easy to use tool for many statistical analyses. This paper deals with design considerations during the development of I.E.S.S., with emphasis on the implementation language and interaction between the PLORTS and Express systems. Examples of usage, sample segments of code, and message file structures are taken from the Correlations-Factor Analysis statistical area.

17. Key Words and Document Analysis. 17a. Descriptors

Interactive Statistics
Time Sharing
Factor Analysis
Computer Software

17b. Identifiers/Open-Ended Terms

Call-OS Fortran
Interactive Express Statistical System

17c. COSATI Field/Group

| 18. Availability Statement | 19. Security Class (This Report) | 21. No. of Pages |
|---|---|---|
| Release Unlimited | UNCLASSIFIED | 53 |
| | 20. Security Class (This Page) UNCLASSIFIED | 22. Price ------- |